UNIVERSITE SAAD DAHLAB Blida 1
Faculté des Sciences

Département d'informatique

THESE DE DOCTORAT

Option : Système Informatique

UNE APPROCHE FORMELLE POUR LA DETECTION D'INTRUSION ET LA CORRELATION D'ALERTES

Par

Nadjah CHERGUI

devant le jury composé de :

H. Bouarfa	Professeur, U. de Blida 1	Présidente
N. Boustia	Professeur, U. de Blida 1	Directrice de thèse
D. Bennouar	Professeur, U. de Bouira	Examinateur
M.Ould khaoua	Professeur, U. de Blida 1	Examinateur
T. Kenaza	Maître de conférences A, E.M.P, Alger	Examinateur

Blida, le 30 novembre 2020

Je dédie ce travail
À ceux qui étaient à mes cotés. . .

REMERCIEMENTS

Au nom de Dieu le Miséricordieux
Alhamdullillah, il m'a donné le courage, la force et la patience pour atteindre ce que je désire.

Je voudrais remercier Mlle. Narhimene Boustia pour tout le soutient et les encouragements qu'elle m'a fournit au cours de mon travail, pour tous ses conseils et ses critiques constructives.

Je tiens à remercier Mm. Bouarfa Hafida, pour m'avoir fait l'honneur d'accepter de présider mon jury de thèse, Je remercie également l'ensemble des membres du jury M. Tayeb Kenaza, M. Djamel Bennouar et M. Mohemed Ould khaoua, qui m'ont fait l'honneur de bien vouloir étudier et examiner mon travail.

J'exprime toute ma gratitude à M.Mohamed Anane qui m'encourage toujours et ne m'a jamais laisser baisser les bras, M. Djamel Bennouar pour tout le parcours qui'il nous a offert durant notre formation GSI. Je dis merci aussi à M. T. Kenaza et M. salem benferhat pour m'avoir accordé du temps pour répondre à mes questions.

Mes remerciements vont également à tous mes collègues du doctorat, spécialement Khadidja Midoun et Khalida Guesmia , avec qui je partage de très bons souvenir.

Ma plus profonde gratitude va à mes parents bien-aimés M. Chergui Belkacem et Mme Guir Oum saad pour avoir toujours été à mes côtés et cru en moi. Ainsi, un grand merci a mes soeurs et frères pour leur amour, leurs prières et leurs encouragements.

Abstract

The security of computer systems is a necessity in the face of attacks that are becoming more and more sophisticated. Unfortunately, the attacks come not only from outside but also from inside the network, hence the need for a well-defined security policy.

Intrusion detection system is an advanced security mechanism, where the majority of organisations deploy IDS in their security policies.

Intrusion detection system requires the intervention of a security administrator to analyze the alerts generated. However, among the limitations of IDSs is the unmanageable number of alerts that includes a high false-positive rate that confuses the security administrator to analyze real attacks.

In this thesis, we propose a formal approach based on the non-monotonic description logic $J\_classic_{\delta\in}$. The proposed approach is based on an ontology that allows contextual information to be represented in order to discover the vulnerabilities exploited by the attacker.

The principle of this approach is that some attacks exploit well-reported vulnerabilities. These last exist in a well defined context where the change of this context implies the disappearance of the vulnerability or the appearance of others. From this point the security administrator must analyze in the first place the relevant alerts.

The approach is evaluated using an instance of the Island hopping attack that was generated to generate network traffic that contains different contexts. Also, the approach is evaluated on the first LLDDOS1.0 scenario of the DARPA 2000 database.

From the observation of the results obtained, our approach represents an effective solution to reduce the number of false positives alerts to deal with the change of the context dynamically. Thus, we consider this model as a complementary tool to the intrusion detection system.

Mots clés :

Intrusion detection system, non-monotonic reasoning, vulnerability, Contextual-ontology

Résumé

La sécurité des systèmes informatiques est une nécessité devant les attaques qui deviennent de plus en plus sophistiquée. Malheureusement, les attaques ne viennent pas seulement de l'extérieur mais aussi de l'intérieur du réseau, d'où la nécessité d'une politique de sécurité bien définie.

Les systèmes de détection d'intrusions forment un mécanisme de sécurité avancée, où la majorité des organisation ont déployé les IDSs dans leur politique de sécurité.

Les IDSs nécessitent l'intervention d'un administrateur de sécurité afin d'analyser les alertes générées. Cependant, parmi les limites des IDSs est le nombre ingérable des alertes qui inclut un taux de faux positifs élevé ce qui empêche l'administrateur de sécurité d'analyser les vraies attaques.

Dans le cadre de cette thèse, nous proposons une approche formelle à base de la logique de description non monotone $J\_\,classic_{\delta\in}$.

L'approche proposée se base sur une ontologie qui permet de représenter les informations contextuelles de façon à découvrir les vulnérabilités exploitées par l'attaquant.

Le principe de cette approche repose sur le fait que certaines attaques exploitent des vulnérabilités bien spécifiques. Ces dernières existent dans un contexte bien défini où le changement de ce contexte implique la disparition de la vulnérabilité ou l'apparition d'autres. De ce point, l'administrateur de sécurité doit analyser sen premier lieu les alertes pertinentes.

L'approche est évaluée en utilisant une instance de l'attaque Island hopping qu'on a généré afin de générer un trafic réseau qui contient différents contextes. En outre, l'approche est évaluée sur le premier scénario LLDDOS1.0 de la base DARPA 2000.

D'après l'observation des résultats obtenus, notre approche représente une solution efficace pour réduire le nombre de fausses alertes permettant de traiter le changement du contexte de façon dynamique. Ainsi, nous considérons ce modèle comme un outil complémentaire au système de détection d'intrusion.

Mots clés :
Système de détection d'intrusion, raisonnement non monotone, vulnérabilité, ontologie contextuelle.

ملخـــــــــــــــص

يعـد أمـن أنظمـة الكمبيوتر ضرورة في مواجهـة الهجمات التي تزداد تعقيـدًا. لسوء الحـظ، لا تأتـي الهجمـات من الخـارج فحسب، بل تأتي أيضًا من داخل الشبكة، ومن هنا تأتي الحاجة إلى سياسة أمنية محددة جيدًا.

نظـام كشـف التسـلل هو آليـة أمنيـة متقدمـة، حيث تقوم غالبية المؤسسـات بتزويـد سياساتهـا الأمنيـة بأنظمـة كشـف التسـلل.

يتطلـب نظـام كشـف التسلل تدخـل مسؤول الأمـن لتحليل التنبيهـات التي تـم إنشاؤها. ومـع ذلك، من بين قيـود أنظمـة كشـف التسلل هو العـدد الذي لا يمكـن التحكـم فيـه من التنبيهـات التي تتضمـن معـدلًا عالـيًا من الإيجابـي الخاطـئ الذي يربك مسـؤول الأمـان لتحليـل الهجمـات الحقيقـية.

فـي هذه الأطروحـة، نقترح نهجًـ رسميًـا يعتمـد على منطـق الوصـف غير الرتيـب $JClassic_{\delta\varepsilon}$, يعتمد النهج المقتـرح على تمثيـل المعلومـات السياقية على شكـل طوبولوجيـا من أجـل اكتشـاف الثغرات التـي يستغلهـا المهـاجم.

مبـدأ هذا النهـج هو أن بعض الهجمـات تستغـل بعض الثغرات. توجـد هذه الأخيرة في سيـاق محدد جيدًا حيث يعنـي تغيير هذا السيـاق اختفاء الثغرات أو ظهور ثغرات أخرى. من هذه النقطـة، يجب على مسؤول الأمـان تحليل التنبيهـات ذات الصلة في المقـام الأول.

يتـم تقييم النهج باستخدام محاكات لهجـوم Island hopping الذي تم إنشـاؤه لإنشاء حركـة مرور شبكيـة تحتوي علـى سياقـات مختلفة. أيضًـا، تم تقييـم النهج باستعمـال السيناريو LLDDOS1.0 الأول لقاعدة بيانـات DARPA 2000.

مـن خلال ملاحظـة النتائـج التي تم الحصـول عليها، يمثل نهجنـا حلاً فعـالاً لتقليل من معدل التنبيهـات الإيجابية الخاطئـة من اجل التعامـل مع تغيرات السيـاق ديناميكيًا. وبالتالـي، فإننا نعتبر هذا النمـوذج أداة تكميليـة لنظـام كشـف التسـلل.

كلمات مفتاحية

نظام كشف التسلل، التفكير غير الرتيب ، الثغرات ، الأنطولوجيا السياقية

Table des matières

Table des figures

Introduction Générale

Contexte et domaine

Les réseaux informatiques deviennent une partie essentielle dans chaque infrastructure. De nos jours, ces réseaux sont souvent reliés au réseau Internet. Cependant, les attaques ne viennent pas seulement de l'extérieur mais aussi de l'intérieur des infrastructures, ce qui nous met devant une grosse masse d'attaques. Par conséquent, il est indispensable de mettre en place une politique de sécurité adéquate.

Au cours des dernières années, différents mécanismes ont été déployés afin de garantir un niveau de sécurité contre les attaques et les tentatives d'intrusions. Parmi ces mécanismes, on s'intéresse aux systèmes de détection d'intrusions (connu comme IDS en anglais pour Intrusion Detection System). En effet, les systèmes de détection d'intrusions sont considérés comme des outils de sécurité indispensables dans la politique de sécurité des infrastructures. Un IDS consiste à analyser des machines, des applications ou des trafics réseau afin de détecter tout signe d'intrusions, dès qu'il détecte une action anormale, il génère un nombre d'alertes qui doivent être analysées par l'opérateur de sécurité pour rapporter les vraies attaques et prendre les contre-mesures.

Cependant, les systèmes de détection d'intrusions souffrent de plusieurs problèmes. Les systèmes de détection d'intrusions génèrent un nombre ingérable d'alertes qui doivent être analysées par l'opérateur de sécurité, ce dernier se retrouve confus pour différencier entre les alertes pertinentes et celle qui correspond aux fausses alertes.En effet, à cause de la similarité entre certaines actions malicieuses et celles de comportement normal, l'IDS considère ces actions comme des connexions légitimes, faux négatifs sont générés en conséquence. En revanche, souvent l'IDS génère un nombre très élevé d'alertes pour des connexions normales, ces alertes représentent des faux positifs. Dans cette thèse, on considère toute alerte générée à cause d'attaques qui n'ont pas pu atteindre leur objectif comme faux positifs.

Ainsi, les attaques deviennent de plus en plus sophistiquées, où l'attaquant réalise une séquence d'actions élémentaires afin d'atteindre son objectif d'intrusion. Et comme les systèmes de détection d'intrusions n'ont pas la capacité de détecter la relation entre ces actions, ils génèrent un ensemble d'alertes pour chacune, ce qui conduit à générer un nombre d'alertes très élevé. Cette lacune était la motivation pour plusieurs chercheurs pour proposer différentes approches de corrélation d'alertes, où l'objectif est de réduire

le nombre d'alertes générées et de regrouper les alertes qui appartiennent au même plan d'attaque.

La corrélation d'alertes est le processus qui consiste à découvrir les relations entre les alertes afin de trouver les alertes pertinentes qui appartiennent au même plan d'attaque. Ce processus est conseillé pour traiter les alertes déclenchées pour des actions élémentaires ou ce qu'on appelle les alertes isolées. Ce type d'approches permet de reconstruire les scénarios d'attaques.

Les connaissances dans le domaine de la détection d'intrusions sont appropriées pour être représentées par des ontologies en appliquant un raisonnement non monotone à cause du changement qui peut se produire dans l'environnement des évènements survenus. Ce changement dû à la configuration du réseau où à des vulnérabilités corrigées.

Les ontologies sont capables de montrer une compréhension partagée d'informations structurées sur les concepts dans un domaine spécifique et fournissent le raisonnement et une plus grande capacité d'analyse automatique des informations. L'utilisation des ontologies spécifiant également les différentes relations sémantiques entre différents concepts, atténuant le problème de l'interopérabilité et la réutilisation.

La non-monotonie est apparue avec les tentatives de représentation d'un raisonnement sur les actions et leur changement. Un raisonnement non monotone peut être décrit comme la théorie de la création et de la sélection des hypothèses de manière dirigée.
Ces dernières sont seulement des hypothèses, donc elles peuvent être abandonnées quand on apprend de nouvelles connaissances sur les circonstances qui provoquent une contradiction avec elles.

En résumé, l'objectif de ce travail est de développer une approche qui permet de réduire le nombre de fausses alertes en appliquant la notion de la non-monotonie pour permettre à l'opérateur de sécurité de sélectionner les alertes pertinentes et élaborer une inférence plus dynamique et flexible.

Problématique

Les systèmes de détection d'intrusions jouent un rôle primordial dans le processus de la sécurité des infrastructures. Ils sont considérés comme une seconde ligne de défense afin de garantir certains niveaux de protection contre les attaques extérieures et celles qui viennent de l'intérieur.

Néanmoins, les systèmes de détection d'intrusions souffrent de plusieurs problèmes, parfois, l'IDS n'arrive pas à détecter certaines attaques qui sont similaires au comportement normal, dans ce cas, on parle de faux négatifs. En revanche, l'IDS considère certaines connexions normales comme une tentative d'intrusions, par conséquent, il génère un ensemble d'alertes qui correspond aux faux positifs. À côté de ces deux soucis, l'IDS en

général, génère un nombre ingérable d'alertes qui doivent être analysées et traitées par l'opérateur de sécurité. En effet, l'opérateur de sécurité se trouve incapable de gérer cette masse d'alertes pour accomplir sa tâche afin de traiter les vraies attaques.

Ainsi, l'attaquant exploite certaines vulnérabilités pour attendre son objectif. Ces vulnérabilités représentent des failles de sécurité dans le système et sont reliées à un contexte particulier qui concerne le type du système d'exploitation utilisé, les ports ouverts, les services, les applications installées et leur environnement spécifique. En effet, une machine est vulnérable si elle satisfait certaines conditions. L'exploitation de ce type d'informations permet d'améliorer la qualité du diagnostique et le taux de détection. En effet, le contexte signifie l'ensemble des informations contextuelles telles la vulnérabilité reliée aux différents services et applications, les utilisateurs, les types de réseau protégés, etc.

Par exemple, une attaque qui exploite une vulnérabilité existante seulement sur une plate-forme Windows, peut déclencher des alertes, même si la cible est un système Unix ou cette vulnérabilité est corrigée contre l'attaque correspondante. Se baser sur les informations contextuelles permet de considérer ces alertes comme faux positifs.

De manière générale, l'ensemble des informations contextuelles donne une vue globale sur les circonstances dans lesquelles l'évènement est survenu. Cela permet à l'opérateur de sécurité d'évaluer et d'analyser les alertes de manière plus efficace pour rapporter les vraies menaces.

Afin d'illustrer l'importance du contexte dans la détection d'intrusions pour réduire le nombre d'alertes non pertinentes, on peut citer l'exemple suivant. Une alerte donnée est générée à cause d'une attaque qui exploite la vulnérabilité (CVE-2017-0143) du service de partage des fichiers Windows qui permet aux attaquants de prendre le contrôle à distance de l'ordinateur. Cette alerte est pertinente si la machine cible exécute ce service et en même temps elle satisfait les conditions nécessaires pour être effectuée par cette vulnérabilité, sinon elle sera considérée comme un non pertinentes

Dans le domaine de la détection d'intrusions, il est intéressant de modéliser le changement de contexte qui peut se produise à cause de diverses raisons telles que la mise à jour de la configuration du système ou bien résoudre les failles des services et applications. Une alerte peut être pertinente dans un contexte mais pas dans un autre quoiqu'elle soit générée à cause de même type d'attaque qui exploite la même vulnérabilité. L'alerte générée est pertinente si la machine cible représente la vulnérabilité exploitée, mais elle n'est pas, si la même attaque cible une machine qui ne satisfait pas les conditions nécessaires de la vulnérabilité exploitée.

Le changement de contexte a un impact sur la sémantique des alertes, ce qui implique un facteur important pour réduire le nombre de faux positifs. La pertinence des alertes est relative au changement des circonstances. La définition des alertes pertinentes doit être faites à chaque changement de contexte, d'où la nécessité d'introduire les connaissances défauts et exceptions.

Formaliser le problème de contexte et ses changements avec les logiques classiques qui supportent un raisonnement monotone pour répondre aux requêtes de pertinence peut provoquer un conflit, de sorte que la conclusion de l'inférence est toujours valide même après l'ajout de nouveaux faits.

Afin de traiter ce conflit, plusieurs travaux sont allés vers la non-monotonie. En effet, le raisonnement non monotone consiste à assurer un raisonnement plus proche au raisonnement de l'être humain. Apprendre de nouveaux faits sur les circonstances peut abandonner le raisonnement qui a été fait avant, où ces nouveaux faits sont une contradiction avec les connaissances 'par défaut'. Les connaissances par défaut sont des faits considérés comme valides dans toutes les circonstances.

Le problème principal traité dans cette thèse est de traiter le nombre trop élevé de faux positifs, en outre, trouver un mécanisme qui permet de modéliser le changement de contexte.

Contributions de la thèse

La corrélation d'alertes permet de trouver la relation entre les alertes pour construire les scénarios possibles d'attaques et de réduire le nombre de faux positifs.

De notre point de vue, prendre en considération les informations sur l'environnement permet de rendre les alertes plus significatives pour que l'opérateur de sécurité puisse analyser les attaques les plus plausibles. Ainsi, profiter de l'expérience de l'opérateur de sécurité par l'interrogation de ses préférences afin de déclarer les sous-réseaux qu'il veut protéger en premier lieu.

A la base d'une étude que nous avons fait et qui montre l'impact et l'importance de la vulnérabilité pour déterminer la pertinence des alertes [CB16], nous avons proposé une approche [CB20] qui vise à associer et enrichir la sémantique des alertes et améliorer la qualité des alertes en se basant sur des informations contextuelles représentées par une ontologie, et aussi d'introduire les préférences de l'opérateur de sécurité dans le processus de la corrélation.

Vu que le contexte détermine la présence ou l'absence de certaines vulnérabilités, donc le changement de contexte peut conduit à un changement sur l'état de la vulnérabilité et qui implique une conclusion différente. Delà l'approche proposée a pour objectif de traiter et modéliser le changement de contexte avec un raisonnement dynamique et flexible.

Comme résultat, notre contribution dans cette thèse permet de réduire le nombre d'alertes non-pertinentes, afin de fournir à l'opérateur de sécurité les alertes qu'il veut l'analyser en premier lieu.

L'évaluation des approches de la corrélation d'alertes dans la détection d'intrusions, s'appuie sur l'utilisation des benchmarks. Dans ce domaine, la majeure limitation de ces

benchmarks manquent d'une description complète de l'environnement et elles ne traitent pas le changement de contexte.

Notre objectif dans cette thèse vise aussi à générer un instance d'attaque qui fournit une description complète du contexte et qui traite le changement de contexte.

Nos travaux mentionnés dans cette thèse peuvent être résumés comme suit :

1. représenter une ontologie à base de contexte qui permet de prendre en considération l'ensemble des informations contextuelles (les types de réseau protégés, les services et les applications installées) et de poser des requêtes de pertinence.
2. Réduire le nombre ingérable d'alertes générées notamment les faux positifs.
3. Générer une instance de l'attaque Island-hopping qui fourni une description sur le contexte du réseau et qui modélise le changement de contexte.

Structure de la thèse

Cette thèse se compose de cinq chapitres au sien de deux parties avec une introduction générale et une conclusion générale.

Dans la première partie (chapitre 1,2 et 3) nous présentons des notion de base sur les systèmes de détection d'intrusion et l'état de l'art.

Dans le chapitre 1, nous présentons un aperçu sur la détection d'intrusions en se focalisant sur les systèmes de détection d'intrusions par la présentation de ses différentes catégories et types.

Le chapitre 2 fourni une étude des différents travaux dont l'objectif est de réduire le nombre d'alertes générées et particulièrement celui de faux positives.

Le chapitre 3 plaide pour la nécessité d'utiliser le contexte et le changement de contexte pour répondre aux besoins de l'opérateur de sécurité afin de garantir une meilleure interprétation des alertes générées ; dans ce chapitre nous illustrons l'utilité de la logique de description $JClassic_{\delta\epsilon}$ qui supporte un raisonnement non monotone dans le domaine de la détection d'intrusions.

Dans la deuxième partie (Chapitre 4, Chapitre 5 et Chapitre 6), nous représentons nos contributions.

Le chapitre 4 présente notre solution pour la corrélation d'alertes et la réduction du faux positifs.

Le chapitre 5 contient l'implémentation de l'ontologie proposée, nous citons quelque problèmes concernant l'évaluation des approches de la corrélation d'alertes et la génération du trafic d'une instance de l'attaque Island-hopping afin d'évaluer l'approche proposée.

Le chapitre 6 montre les expérimentations de l'approche proposée. Dans ce chapitre, nous discutons les résultats obtenus on se base sur le nombre de faux positifs réduit et l'efficacité de traiter le changement de contexte.

Nous terminons cette thèse par une conclusion générale qui résume l'ensemble des contributions, ainsi qu'une suggestion pour les recherches futures.

Première partie

Les systèmes de détection et la corrélation d'alertes

Chapitre 1

Connaissances de base sur la détection d'intrusions

1.1 Introduction

La vulnérabilité des systèmes informatiques représente un problème sérieux devant les administrateurs de sécurité. En effet, un attaquant ayant obtenu un accès sur un système ou un réseau peut alors effectuer la violation des informations afin d'atteindre ses objectifs. Les systèmes de détection d'intrusions (noté IDS en anglais : Intrusion Detection System) jouent un rôle primordial dans le processus de la sécurité informatique. Un IDS consiste à découvrir toute tentative d'accès non autorisée au réseau ou bien au système informatique en analysant les informations collectées. Il effectue une détection automatisée qui correspond à la génération d'un ensemble d'alertes. Ces dernières doivent être traitées et analysées par l'administrateur de sécurité.

La détection d'intrusion est défini comme le problème de l'identification des personnes qui utilisent un système informatique sans autorisation et de celles qui ont un accès légitime au système mais abusent de leurs privilèges. De cela, la détection d'intrusion consiste à surveiller les événements qui se produisent dans un système informatique ou un réseau et les analyser pour rechercher d'éventuels incidents qu'il s'agisse des violations des stratégies de sécurité informatique, des politiques d'utilisation acceptables ou des pratiques de sécurité standard [SM12].

Dans ce chapitre, nous présentons d'abord, quelques définitions sur les notions de base de la détection d'intrusion. Ensuite, nous entamons les systèmes de détection d'intrusions en détails, i) on présente le fonctionnement des IDS, ii)la classification des systèmes de détection d'intrusions. iii)les différents types de détection d'intrusions, iv)les catégories des systèmes de détection d'intrusions.

1.2 Définitions de base

Dans cette section, nous présentons quelques définitions concernant le domaine de la détection d'intrusion.

Définition 1.2.1 (Intrusion) *C'est la prise de contrôle partielle ou totale d'un système distant par l'exploitation d'une vulnérabilité bien précise.*

Définition 1.2.2 (Attaque) *C'est l'exploitation des vulnérabilités des systèmes informatiques (système d'exploitation, logiciel ou bien même de l'utilisateur) afin de briser la politique de sécurité, généralement, il peut-être défini par toute action qui compromet la sécurité des informations.*

Définition 1.2.3 (Attaque complexe) *C'est un type d'attaque qui se réalise à travers plusieurs étapes où un ensemble d'actions élémentaires.*

Définition 1.2.4 (objectif d'intrusion) *C'est le résultat d'une intrusion qui permet de compromettre le système informatique, il se trouve à la fin du processus de l'attaque.*

Définition 1.2.5 (Scénario d'attaque/ plan d'attaque) *C'est l'ensemble d'actions exécutées par l'attaquant afin d'atteindre l'objectif d'intrusion.*

Définition 1.2.6 (Vulnérabilité) *Une vulnérabilité est une faute accidentelle ou intentionnelle (avec ou sans volonté de nuire), dans la spécification, la conception ou la configuration du système, ou dans la façon selon laquelle elle est utilisée. La vulnérabilité peut être exploitée pour créer une intrusion [PS03].*

Définition 1.2.7 (Vulnérabilité exploitée) *l'attaquant utilise une vulnérabilité bien spécifique afin de réaliser son objectif, elle peut être présente dans le système cible ou absente.*

Définition 1.2.8 (Alerte) *c'est un message généré par le système de détection d'intrusion.*

Définition 1.2.9 (Détection d'intrusion) *C'est le processus pour identifier et atténuer les attaques en cours à l'aide d'un système de détection d'intrusion.*

Définition 1.2.10 (Administrateur de sécurité) *C'est le responsable qui s'occupe des équipements de sécurité et qui gère les alertes des systèmes de détection d'intrusion.*

Définition 1.2.11 (Politique de sécurité) *C'est un ensemble de mesures déployées pour préserver la sécurité du système informatique.*

1.3 Rôle des systèmes de détection d'intrusions

Les systèmes de détection d'intrusions jouent un rôle primordial dans les différentes infrastructures des organisations pour atteindre l'objectif de protéger l'information contre toute tentative de violation [SM12].

Dans ce qui suit, nous présentons les différentes raisons pour lesquels les systèmes de détection d'intrusions ont été intégré dans le processus de la sécurité informatique.

Le rôle majeur des IDSs est l'identification des événements malicieux et rapporter ces évènements à l'administrateur de sécurité pour prendre une décision concernant cette intrusion. Pour faire ça, ils génèrent un ensemble de messages appelés alertes, ces dernières contiennent des différentes informations qui pourraient être utilisées par les gestionnaires d'incidents.

— Les systèmes de détection d'intrusions peuvent être utilisés pour identifier les problèmes de sécurité dans un autre système afin de contrôler le fonctionnement des outils de sécurité telle que la duplication des règles de pare-feu et générer des alertes dans le cas où il voit un trafic réseau qui pourrait être bloqué par le pare-feu, mais pas à cause d'une erreur de configuration [SM12].

— Même si les IDSs ne sont pas en mesure de bloquer les attaques, ils peuvent toujours collecter des informations sur les attaques. Ces informations peuvent être pertinentes pour protéger la politique de sécurité de l'organisation.

— En général, les attaques ne proviennent pas toujours de l'extérieur, souvent, les attaques proviennent de l'intérieur du réseau par des utilisateurs qui voulait gagner un accès non autorisé. Mettre en place des IDSs peut influencer sur le comportement des utilisateurs individuel et protéger le système en augmentant le risque de découverte et de punition des attaquants.

— Un IDS peut fournir des avertissements indiquant qu'un système est attaqué même si le système n'est pas vulnérable à l'attaque spécifique. Ces avertissements peuvent aider les utilisateurs à modifier la posture défensive de leur installation pour augmenter leur résistance aux attaques [MCA00].

1.4 Classification des systèmes de détection d'intrusions

Vu l'évolution des systèmes de détection d'intrusions, ils sont classifiés selon un nombre de critères comme il est illustré dans la figure 1.1.

1.4.1 Méthode d'analyse

On distingue deux approches importantes selon la méthode d'analyse utilisée pour distinguer entre les activités normales (légitimes) et les activités malicieuses.

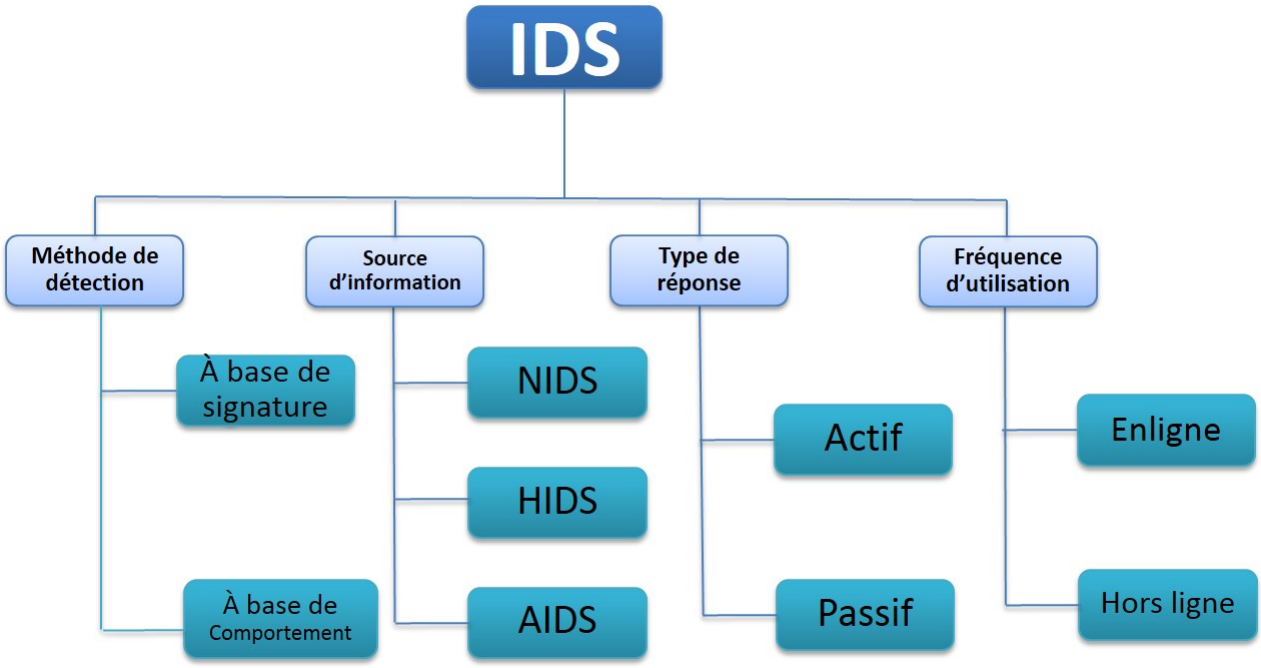

FIGURE 1.1 – Taxonomie des systèmes de détection d'intrusions [DDW99]

La première approche est dite par comportement, elle consiste à utiliser des informations sur le comportement du système analysé. L'autre approche est dite par signature, elle consiste d'utiliser des informations sur les attaques. Une signature est un motif qui correspond à une menace connue.

— Approche par comportement : cette approche consiste à définir un profil normal qui décrit le comportement normal du système et des utilisateurs. Ce profil est développé en surveillant les caractéristiques des activités du système sur une période de temps. Les profils peuvent être développés pour de nombreux attributs comportementaux, tels que le nombre d'e-mails envoyés par un utilisateur, le nombre de tentatives de connexion échouées pour un hôte et le niveau d'utilisation du processeur pour un hôte sur une période donnée. A partir de là, toute activité étrangère par apport au profil normal est considérée comme un comportement malveillant, ce qui implique que l'IDS doit générer un ensemble d'alertes pour rapporter l'intrusion à l'administrateur de sécurité.

  Afin de définir le comportement normal, deux techniques sont utilisés : les statistiques et l'intelligence artificielle [Ken11]. L'avantage de cette approche est sa capacité de garantir la détection de toute tentative d'intrusion même si l'attaque est nouvelle et elle n'est pas encore intégrée dans les bases d'attaques prédéfinies. Ainsi, elle est efficace pour détecter les nouvelles et les vulnérabilités imprévues pour faciliter la détection d'abus de privilège. Néanmoins, cette approche implique la généralisation

d'un nombre important de faux positifs dans le cas du changement du comportement qui peut se produise, ainsi le système d'information peut subir des attaques en même temps que le système de détection d'intrusions apprend. De même la précision des profils est faible en raison d'événements constamment modifiés.

— Approche par signature : cette approche repose sur une base d'attaques prédéfinies par des systèmes d'experts qui contient un ensemble de règles pour décrire les attaques [WSP08]. Elle consiste à détecter les signatures des attaques en les comparant avec ceux prédéfinis. Autrement dit, toute action qui n'est pas explicitement reconnue comme une attaque est considérée comme acceptable. L'avantage de cette approche est de garantir la détection des attaques connues. Néanmoins, elle risque de ne pas observer les nouvelles attaques ou les variantes des attaques connues qui peuvent amener un grand danger dans le système analysé. Pour cela, il est nécessaire de mettre à jour fréquemment la base des attaques.

1.4.2 Source d'audit

On distingue différents types de technologies pour les systèmes de détection d'intrusions. On les différencie selon le type des évènements qu'ils analysent et la façon dont ils sont déployés. Certains analysent le trafic réseau, d'autres analysent les évènements qui se produisent sur une machine et applications.

— IDS basé sur l'hôte (HIDS) : système de détection d'intrusions basé hôte, il est déployé sur une machine unique pour surveiller les évènements se produisant dans cette machine comme le trafic réseau, les fichiers logs du système, les processus en cours d'exécution, les activités des applications, l'accès et la modification des fichiers et les modifications de la configuration du système et d'application. Les HIDS sont souvent déployés sur des machines critiques tels que des serveurs accessibles au public et des serveurs contenant des informations sensibles.

— IDS basé sur le réseau (NIDS) : système de détection d'intrusions basé réseau, il surveille le trafic réseau pour un réseau particulier et analyse les protocoles de réseau, de transport, et d'application afin d'identifier les activités suspectes. Il se compose de plusieurs capteurs, un ou plusieurs serveurs de gestion, plusieurs consoles, et éventuellement, un ou plusieurs serveurs de base de données. Souvent, les capteurs sont déployées en ligne pour que le trafic réseau qu'il surveille les traverses, tout comme le flux de trafic associé à un pare-feu. Le déploiement des capteurs en ligne permet d'arrêter les attaques en bloquant le trafic réseau.

— IDS à base d'application (AIDS) : ce type de système de détection d'intrusion vise à surveiller les fichiers log de certaines application dans un hôte.

1.4.3 Comportement de détection

Un système de réponse devrait être intégré au système de détection d'intrusion pour aider et trouver la source d'une attaque [KGWB15]. Un IDS est classé comme passif ou actif selon l'option de réponse.

— Passif : lorsqu'un IDS ne prend aucune contre-mesure lors de la détection d'une intrusion. Il se contente de générer des rapports et des alertes.

— Actif : On dit qu'un IDS est actif, lorsqu'il répond immédiatement à une intrusion et prend des mesures correctives ou préventives sans l'intervention de l'homme. Cette approche peut prendre comme contre-mesure : bloquer un compte utilisateur, terminer une session, éteindre l'ordinateur, désactiver le port ou le service attaqué, etc [YÇ10].

1.4.4 Mode d'exploitation

En terme de fréquence d'utilisation, les systèmes de détection d'intrusions peuvent être catégorisés en deux catégories principales :

— En ligne : un IDS en ligne effectue une analyse des activités réseau en temps réel, c-à-d, il fait un traitement des évènements surveillés lors de leur apparition. L'avantage principal de cette approche est que les activités du système peuvent être analysées en temps opportun. Ainsi, une réponse appropriée peut être émise lorsqu'une attaque est détectée. Cependant, les frais généraux du système dans ce cas sont significativement élevés.

— Hors ligne : un système de détection d'intrusions hors ligne effectue une analyse des évènements réseau et les activités du système de manière périodique dans des intervalles de temps réguliers. Par conséquent, en cas d'attaque réussie, ils ne peuvent être utilisés que pour l'analyse post-mortem.

1.5 Efficacité des systèmes de détection d'intrusions

Les chercheurs ont proposé un certain nombre de métriques pour évaluer l'efficacité d'un système de détection d'intrusion. Ces métriques permettent de choisir le système de détection approprié selon les besoins de l'application. Dans ce qui suit, nous citons un ensemble de critères proposés par les chercheurs dans le travail [DDW00] et [ER08].

— Performance : la performance d'un IDS dépend du taux de traitement des évènements d'audit. Un IDS est dit performant si la détection en temps réel est possible.

— Complétude : la complétude indique que l'IDS est capable de détecter toutes les attaques. En revanche, l'incomplétude se produit quand un IDS manque d'analyser une

attaque particulière. L'évaluation selon ce critère est une tâche difficile et nécessite plus de détails sur l'attaque.

— Disponibilité : un IDS devrait lui-même résister aux attaques, en particulièr à un déni de service. Ce critère est particulièrement important, car la plupart des systèmes de détection d'intrusions fonctionnent sous des systèmes d'exploitation ou des matériels disponibles dans le commerce, qui sont connus pour être vulnérable aux attaques.

— Rapidité : un IDS devrait être capable d'effectuer le traitement des évènements aussi vite que possible. Ce critère est important pour les systèmes de détection d'intrusions, car la plupart risquent de manquer quelques paquets.

1.6 Les composants de système de détection d'intrusion

Dans ce qui suit, nous décrivons les composants majeurs des systèmes de détection d'intrusions. Puis, nous illustrons les différentes architectures pour déployer ces composants dans le réseau.

— Capteur : un capteur surveille et analyse les activités qui se produisent dans un réseau. Les capteurs pour les systèmes de détection basé hôte sont connues sous le mot agent.

— Serveur de gestion : est un dispositif centralisé qui reçoit et gère les informations à partir des capteurs. Certains serveurs de gestion peuvent identifier les évènements que les capteurs individuelles ne peuvent pas détecter. Delà, la notion de la corrélation peut être connue comme la correspondance des informations reliées aux évènements de plusieurs capteurs.

— Serveur de base de données : est un dépôt pour enregistrer les informations des évènements fournis par les capteurs et les serveurs de gestion.

— Console : c'est un programme qui fournit une interface pour les utilisateurs et les administrateurs d'IDS. Le logiciel de console est généralement installé sur des ordinateurs de bureau ou portables standard. Certaines consoles sont uniquement utilisées pour l'administration d'IDS, telles que la configuration de capteurs ou d'agents et l'application de mises à jour logicielles, tandis que d'autres consoles sont strictement utilisées pour la surveillance et l'analyse. Certaines consoles IDS fournissent des capacités d'administration et de surveillance.

1.7 Problèmes liés aux systèmes de détection d'intrusions

Les systèmes de détection d'intrusion analysent les évènements survenus dans un système informatique ou bien dans un réseau informatique ; pour découvrir les évènements illégaux. Ces dernières se produisent pour diverses raisons, telles que les malwares, les attaquants qui obtiennent un accès non autorisé aux systèmes à partir d'internet et les utilisateurs autorisés qui abusent de leurs privilèges tentent d'obtenir des privilèges supplémentaires pour lesquelles ils ne sont pas autorisés. Bien que de nombreux évènements soient de nature malveillantes, beaucoup d'autres ne sont pas ; par exemple une personne pourrait fausser l'adresse d'un ordinateur et essayer accidentellement de se connecter à un système différent sans autorisation [SM12].

Quoique les systèmes de détection d'intrusion aient gagné l'acceptation d'être un outil indispensable dans la politique de sécurité, ils sont encore incapables de détecter toutes les attaques complexes. En outre, et parfois, les systèmes de détection d'intrusion considèrent des faits normaux comme des faits anormaux, cela implique la génération de fausses positives. En revanche, les IDSs peuvent rater la détection d'une activité malicieuse ce qui implique de fausses négatives.

Ainsi, les systèmes de détection d'intrusion génèrent un nombre très élevé d'alertes où la majorité sont de fausses alertes.

L'une des raisons pour lesquels un IDS génère un pourcentage élevé de fausses positives est qu'il ne prend pas en considération les informations contextuelles qui caractérisent les évènements survenus. Les informations contextuelles déterminent les vulnérabilités liées aux applications, aux services, aux utilisateurs, et à les types de réseau protégés. Delà, l'administrateur de sécurité doit intervenir pour analyser les alertes afin de rapporter les vraies attaques et ignorer ceux qui n'ont pas réussi à obtenir leurs objectifs d'intrusion.

Prendre en compte les informations contextuelles pour vérifier la pertinence des alertes générées, implique que toutes les alertes qui sont générées pour des attaques non réussites, sont de faux positifs et non pertinentes.

En résumé, la majorité des problèmes liés aux systèmes de détection d'intrusion se résument dans les points suivants :

— Une grande masse d'alertes : les IDSs génèrent un nombre très élevé d'alertes pour chaque évènement détecté sans aucune description. Ce fait-là rend la tache de l'administrateur de sécurité plus difficile afin d'analyser cette masse d'alertes.

— Le taux de faux positifs et faux négatifs : l'un des problèmes liés aux systèmes de détection d'intrusion est le grand pourcentage de fausses alertes générées (faux positifs et faux négatifs) qui complique la tâche de l'administrateur de sécurité beaucoup plus pour identifier les vraies attaques.

— Les alertes non pertinentes : le fait que les IDSs sont incapables de vérifier le contexte des alertes générées, ça implique la génération d'un ensemble d'alertes non pertinentes même si elles correspondent à des vraies intrusions, ces intrusions n'ont pas réussi à réaliser leurs objectifs.

1.8 Conclusion

Les systèmes de détection d'intrusion sont considérés comme un outil indispensable dans la politique de sécurité des infrastructures. Les IDSs représentent une deuxième ligne de défense où ils doivent être couplés à plusieurs autres mécanismes afin de garantir une sécurité optimale.

Le premier rôle des systèmes de détection d'intrusions est de détecter toute tentative qui vise à contourner la politique de sécurité pour compromettre les vulnérabilités des systèmes. Par conséquent, ils générèrent un ensemble d'alertes qui doivent être vérifiées par l'administrateur de sécurité.

On peut classer les systèmes de détection d'intrusions en différentes catégories selon un ensemble de critères qui ont un impact sur le fonctionnement d'IDS. Parmi ces critères nous avons cité la méthode d'analyse qui peut spécifie le comportement d'un IDS pour détecter les intrusions, source d'audit qui précise le type des fichiers à analyser, le comportement de détection qui précise l'option de réponse en cas de détection des intrusions et la fréquence d'utilisation qui indique le fonctionnement en temps réal.

Cependant, les systèmes de détection d'intrusions souffrent de plusieurs problèmes. La génération d'un énorme volume d'alertes avec un taux de faux positifs très élevées est parmi les problèmes les plus importants qui font face aux systèmes de détection d'intrusions. En outre, les IDSs ne prennent pas en considération l'ensemble des informations contextuelles qui caractérisent l'environnement des alertes générées afin de vérifier la pertinence de ces alertes.

Tenant compte de l'évolution rapide des systèmes et technologies avec à chaque fois un lot de nouvelles vulnérabilités. Il est nécessaire de proposer et de développer des solutions pour améliorer le fonctionnement d'un IDS.

Nous nous focalisons dans le chapitre suivant sur les différentes approches proposées dans la littérature pour atteindre l'objectif de réduire le nombre des alertes générées en particulière le taux de faux positifs, en détaillant leurs principes, leurs valeurs ajoutées, et leurs limites.

Chapitre 2

Techniques de corrélation d'alertes

2.1 Introduction

Les systèmes de détection d'intrusion sont largement répandus de nos jours pour la sécurité des systèmes informatiques. Ils permettent à la fois de détecter et de répondre aux attaques en temps réel ou en hors-ligne.

Avec le développement de la technologie, les attaques sont devenues plus sophistiques et plus complexes. Une attaque peut se réaliser sous forme de plusieurs phases en exploitant les failles du système. De ce fait, les systèmes de détection d'intrusions génèrent un énorme volume d'alertes avec un taux de faux positif très élevé, faisant de leurs analyses une tâche difficile pour l'opérateur de sécurité qui tente de déterminer et d'identifier les alertes pertinentes. En outre la plupart des systèmes de détection d'intrusions ne prennent pas en considération l'ensemble des informations contextuelles qui caractérisent l'environnement des évènements survenus. Par conséquent, les systèmes de détection d'intrusions génèrent une grande partie des alertes non pertinentes même si ces alertes correspondent à des vraies attaques. dû au fait que ces attaques n'ont pas réussi à atteindre leurs objectifs.

Les limites mentionnées dans le paragraphe précédent représentent la motivation de plusieurs travaux dans la littérature, dont l'objectif est de réduire le nombre des alertes générées ainsi que de réduire le taux de faux positif et alertes non pertinentes. Yusof et al. [YSS08] ont classé les travaux qui traitent la corrélation des alertes et la détection d'intrusions en quatre catégories. Cette catégorisation est faite selon les techniques utilisées pour réduire le nombre d'alertes : techniques à base de similarité, techniques à base de scénario d'attaques prédéfinis, techniques à base filtrage et des techniques de corrélation d'alerte multi-steps. On peut ajouter aussi une autre catégorie qui prend en considération des informations contextuelles pour interpréter les alertes.

Cependant, la plupart de ces travaux n'ont pas pris en considération le changement de contexte qui peut se produire fréquemment. Notre travail est basé sur ces travaux. On va présenter le contexte des alertes générées de manière dynamique et flexible tout en prenant en compte le changement de contexte. Dans ce chapitre, nous nous présentons en détail un état de l'art des techniques de corrélations d'alertes et nous nous focalisons sur les travaux

qui traitent le contexte des alertes générées.

2.2 La corrélation d'alertes et la détection d'intrusions

Le mot corrélation apparaît souvent dans les travaux qui traitent la détection d'intrusions. Dans cette section, nous présentons une étude sur les différentes approches de la corrélation d'alertes en commençant par quelques définitions.

Définition 2.2.12 (Définition de la corrélation d'alertes) *La corrélation d'alerte est un processus en plusieurs étapes qui reçoit des alertes d'un ou plusieurs IDS comme entré et produit une description de haut niveau de l'activité malveillante sur le réseau [MAJ13]. Selon les auteurs dans [Dan03], la corrélation d'alerte fait référence à l'interprétation, la combinaison et l'analyse d'alertes, ainsi qu'à des informations externes au système de détection d'intrusion, dont le but est de réduire le volume d'alertes en reconstruisant les scénarios d'attaques, et améliorer potentiellement les performances de détection grâce au renforcement des capteurs ou à la complémentarité.*

2.3 Objectifs de la corrélation d'alertes

La corrélation d'alertes est la solution pertinente pour traiter les problèmes liés au volume excessif d'alertes. Dans cette section, nous présentons les objectifs principaux de la corrélation d'alertes.

2.3.1 Réduire le volume d'informations

Souvent, les systèmes de détection d'intrusions produisent un énorme nombre d'alertes qui correspondent à des évènements spécifiques. La corrélation d'alertes vise à réduire le volume d'informations générées en appliquant différentes techniques, parmi elles :

- L'élimination des alertes redondantes qui peuvent être générées pour un seul évènement, dans ce cas, la corrélation d'alerte vise à garder qu'une seule de ces alertes.
- La fusion : pour simplifier le traitement des informations des alertes qui correspondent à des attaques complexes, il est préférable de fusionner les informations qu'elles contiennent.
- L'agrégation des alertes : regrouper les alertes en sous ensembles appelés clusters en se basant sur les attributs communs (l'adresse de l'attaquant, l'adresse de la cible, le type de l'attaque) de ces alertes.

— La synthèse : cette technique vise à trouver une relation entre les alertes comme par exemple des règles logiques qui peuvent lier les alertes à des scénarios d'attaque connues, ou des lois statistiques apprises. Cette technique peut synthétiser un ensemble d'alertes en une alerte résumant l'ensemble des connaissances.

2.3.2 Améliorer la qualité du diagnostic

Les alertes générées contiennent des informations de bas niveau et manquent de sémantique. Ces alertes ne fournissent pas les informations suffisantes sur la vulnérabilité exploitée.
Les techniques utilisées pour réduire le nombre d'alertes peuvent être utilisées pour améliorer la qualité du diagnostic et la qualité des alertes.

2.3.3 Suivre les attaques

Souvent, les attaques se composent de plusieurs étapes afin de collecter des informations sur le système cible, les services publiés et les composants physiques et logiques du réseau. L'attaquant utilise ces renseignements pour préparer son attaque et compromettre la politique de sécurité du système cible.
Ce fait implique la génération de plusieurs alertes. Delà, l'objectif de la corrélation d'alertes est de comprendre la stratégie de l'attaque en identifiant les actions de l'intrusion. La corrélation d'alertes doit permettre d'archiver et de classifier les actions réalisées, les informations accumulées sur l'attaquant et les informations que les cibles fournissent.

2.4 Techniques de corrélation d'alertes

La corrélation d'alertes consiste à mettre en évidence des relations entre les alertes sans schéma préétabli. Afin de réduire le nombre de faux positifs, plusieurs approches de corrélation sont proposées. Ces travaux sont classés en quatre catégories : des approches qui reposent sur des mesures de similarité entre les attributs des alertes, des approches qui se basent sur la reconstruction des scénarios d'attaques, des approches à base de filtrage en donnant des priorités à des alertes particulières et des approches qui consistent à découvrir les attaques multi-steps.
Dans ce qui suit, nous détaillons chaque catégorie en mentionnant ses principaux apports.

2.4.1 Corrélation d'alertes à base de similarité

Les approches à base de similarité corrèlent les alertes en se basant sur la similarité entre les alertes. Le calcul de la similarité repose sur une définition de similarité entre les alertes. La mesure de similarité permet d'agréger les alertes, c'est-à-dire effectuer des

regroupements d'alertes jugées plus ou moins proches en fonction de leurs attributs. Les fonctions de similarité entre les alertes sont des combinaisons de similarité définies sur leurs attributs. Les fonctions de similarité sur les attributs se basent sur des connaissances d'experts liées aux attaques et à l'environnement. Dans ce genre d'approches, il est important de spécifier deux issues : les attributs sur lesquels l'approche se base et la méthode d'évaluation.
A-Valdes et al. [VS01] ont proposé une approche probabiliste pour la corrélation d'alertes. Cette approche se base sur les attributs que les alertes contiennent pour appliquer un algorithme de fusion afin de corréler les alertes qui semblent similaires et proches. L'algorithme de fusion se concentre seulement sur les attributs communs où un seuil est utilisé pour déterminer la correspondance entre ces attributs. Le seuil est contrôlé par un seul paramètre configurable.
Pour fusionner les alertes similaires, les auteurs ont proposé de construire des métas-alertes, chacune contient des listes de valeurs des attributs communs entre les alertes fusionnées. Chaque nouvelle alerte est comparée avec les métas-alertes existants. Dans ce cas, on distingue deux possibilités :

— La nouvelle alerte se fusionne avec la méta-alerte la plus proche à condition que la similarité soit jugée suffisante.
— La nouvelle alerte ne correspond à aucune méta-alerte, ce fait-là implique la création de nouvelle méta-alerte.

La fusion est l'union des valeurs prises par les attributs des alertes contenus dans une méta-alerte. Cette opération se fait en complétant chaque attribut de la méta-alerte par l'attribut de l'alerte à fusionner. Afin de fusionner les alertes, des fonctions de similarité spécifiques ont été définies. Ces fonctions considèrent les issues suivantes :

— Comment spécifier que deux listes soient communes, par exemple la liste des ports cibles.
— Lorsque deux adresses source sont différentes, est-ce qu'elles appartiennent au même sous-réseau.

Afin de déterminer la similarité entre les classes d'attaques, une matrice de similarité a été maintenue par des experts.

Dans cette solution, un ensemble de contraintes de similarité a été identifié. Certains attributs nécessitent une correspondance totale avec une similarité minimum égale à 1 ou bien approximative avec un minimum de similarité moins de 1 mais strictement positive, la similarité entre deux alertes est 0 si tout attribut commun a une correspondance moins que la similarité minimum pour cet attribut. Ces conditions sont nécessaires pour dire qu'une alerte peut être considérée comme un candidat de fusion avec un autre.

Les attributs qui sont tenus en compte sont la source de l'attaque, la cible (machine, numéro de port), la classe d'attaque, la sonde et les informations de temps. Pour chaque attribut, une fonction de similarité appropriée est associée, cette fonction retourne une valeur entre zéro et un sachant que la valeur 1 correspond une similarité parfaite. Ces fonctions de similarité sont utilisées pour fusionner les alertes reliées entre eux.

La fusion se fait par compléter chaque attribut de la méta-alerte par l'attribut de l'alerte à fusionner. La fonction de similarité entre une alerte Y et une *méta-alerte* X est définie comme la moyenne pondérée des mesures de similarité entre les attributs communs de X et Y

$$Sim(Y,X) = \frac{\sum_j E_j sim_j(Y_j,X_j)}{\sum_j E_j}$$

Avec : X= la candidate méta-alerte à fusionner.

Y= la nouvelle alerte.

J= indice pour parcourir les attributs de l'alerte.

E_j= la pondération de l'attribut j.

X_j= une liste de valeurs prises par l'attribut Y_j des alertes participant à la méta-alerte X.

Y_j = la valeur de l'attribut j de l'alerte Y

La fonction *Sim* est la fonction de similarité associée à l'attribut j, qui retourne une valeur entre $[0\,;\,1]$. Les fonctions de mesure de similarité sim_j entre les attributs prennent en compte des caractéristiques propres à chaque attribut. Par exemple, deux adresses IP sources sont similaires si elles appartiennent au même sous-réseau. Les valeurs de similarité entre classes d'attaques sont contenues dans une matrice carrée asymétrique fournie par un expert. Deux attaques seront jugées proches si elles sont identiques, ce qui dénote plusieurs tentatives visant un même objectif ou cohérent, c'est-à-dire que les effets d'une attaque permettent d'effectuer l'autre attaque. La matrice est asymétrique, car la similarité entre deux classes d'attaques distinctes dépend de l'ordre d'occurrence des attaques. Par exemple, comme il est plus logique pour un attaquant d'acquérir d'abord l'information sur sa cible, puis de l'attaquer, la similarité entre un scan de ports suivis d'un déni de service est plus importante que dans l'ordre inverse.

Les pondérations E_j utilisées dans le calcul de similarité des alertes symbolisent la similarité attendue des attributs. La pondération permet d'accroître ou au contraire de mitiger l'importance d'un attribut dans le calcul de similarité d'une alerte avec la méta-alerte. Cette pondération est en fonction du contexte, qui inclut des informations liées aux autres attributs de l'alerte considérée, mais aussi des alertes précédemment impliquées dans la méta-alerte. Par exemple, dans le cas d'une attaque dont la source peut être forgée, le poids associé à la mesure de similarité de la source est faible.

En plus des pondérations utilisées pour diminuer ou accroître l'influence de certains paramètres, le calcul de similarité est contraint par des seuils de similarité minimaux sur

les attributs. Si la similarité entre deux attributs est inférieure au seuil minimal, alors la similarité entre l'alerte et la méta-alerte est nulle. Les seuils permettent d'influer sur la nature des métas-alertes formés : en diminuant le seuil qui contrôle la similarité de la sonde génératrice de l'alerte, les métas-alertes peuvent fusionner des alertes issues de sondes hétérogènes ; en diminuant le seuil correspondant à la classe d'attaque, les métas-alertes peuvent contenir les alertes relatives aux différentes étapes d'une attaque composée.

B-Cuppens [Cup01] propose une technique pour corréler les alertes similaires en se basant sur des règles logiques. Cette approche se déroule en deux étapes. La première consiste à agréger et regrouper les alertes similaires afin de créer des clusters, la deuxième étape de synthétiser les alertes de ces groupes en créant des relations entre les alertes de même cluster. La similarité des alertes dans la phase d'agrégation est une combinaison de la similarité des attributs. Pour chaque attribut, des règles de similarité sont définies où les attributs qui sont tenus en compte sont la source de l'attaque, la cible et le temps. Afin d'identifier les attaque, Cuppens utilise l'identifiant d'attaque fournie par les sondes. Comme les constructeurs d'IDS utilisent des identifiants propriétaires, la similarité de l'attribut correspondant au type d'attaque est définie à l'aide d'un référentiel commun établi dans le cadre du projet MIRADOR3. Ce référentiel est comparable à la liste CVE, qui contient les identifiants des vulnérabilités connues : deux identifiants sont jugés similaires s'ils font référence à la même entrée dans la liste.

Selon le jugement de similarité, la nouvelle alerte est intégrée dans des clusters existants, le prédicat *cluster_ alert(cluster id ,alertid)* spécifie que l'alerte identifiée par *alertid* appartient au cluster identifié par clusterid. Sinon, un nouveau cluster sera créé.

Dans cette approche, une alerte peut appartenir à plusieurs groupes. Ce fait-là implique que des groupes indépendants peuvent être eux-mêmes agrégés suite à l'intégration d'une alerte commune.

Après le regroupement des alertes, le processus de synthèse se déclenche. La synthèse consiste à créer une alerte globale pour chaque groupe d'alertes. La liste des attributs de cette alerte globale est l'union des valeurs des attributs des alertes du groupe correspondant.

C-Xu et al. [XN05] ont proposé une approche de corrélation pour préserver la confidentialité qui peut être dans les alertes. L'approche est proposée dans le contexte de la coopération entre différentes organisations contre les attaques ; où le besoin d'équilibrer les exigences de confidentialité et le besoin d'analyse d'intrusion sont pris en compte.

Plusieurs organisations telles que CERT Coordination Center et DShield [1] collectent des données (y compris des données sur les incidents de sécurité) sur Internet, effectuent des analyses de corrélation et diffusent des informations aux utilisateurs et aux fournisseurs.

1. (http :/www.dshield.org/)

Les données sur les incidents de sécurité sont généralement collectées auprès de différentes sociétés, organisations ou individus, et leurs préoccupations en matière de confidentialité doivent être pris en compte.

L'approche est basée sur la hiérarchie des concepts, elle se déroule sur deux phases. La première phase consiste à affiner les attributs sensibles des alertes, ces attributs sont généralisés aux concepts de haut niveau pour introduire une incertitude dans l'ensemble des données avec une sémantique partielle.

La deuxième phase est la corrélation des alertes affinées en utilisant des fonctions de similarité pour construire des scénarios d'attaques. Les auteurs utilisent des heuristiques pour calculer la similarité entre les attributs catégoriques affinés. Si x_0 et y_0 sont deux attributs d'origine connus, la similarité entre eux est donnée comme suit :

$$sim(x_0, y_0) = \begin{cases} 1 & si\ x_0 = y_0 \\ 0 & sinon \end{cases}$$

Après l'affinement, x_0 et y_0 deviennent des valeurs généralisées x_g et y_g , respectivement. Afin de calculer la similarité entre x_g et y_g. Dans une hiérarchie conceptuelle, le calcule de la similarité se base sur la probabilité que les noeuds (X_g) et (Y_g) ont le même chemin de la racine à une feuille. Si la probabilité que X_g et Y_g aient la même valeur d'origine qui est grande, ils s'interprètent comme une haute similitude entre eux ; sinon leur similarité est faible.

Une recherche récente[HSH19] vise à découvrir la similarité entre les alertes sans avoir des connaissances préalables. Après l'extraction de certains attributs des flux d'alertes sachant l'adresse IP source, l'adresse IP destination, port source, pour destination, protocole, longueur de port, et le nouvel attribut extrait des alertes s'appelle count qui est un compte obtenu à partir de l'apparition du même type d'alerte en fonction des attributs sélectionnés. Deux types d'analyse se faites afin de corréler les alertes en fonction de leur similarité, une analyse de bas-niveau qui détermine la similarité entre les attributs sélectionnés de sort que la similarité entre les adresses IP prend une valeur entre 0 et 1, elle est calculée en fonction du nombre de bits similaires et la longueur de l'adresse IP, cependant, la similarité entre les autres attributs prend comme valeur 0 ou 1. L'analyse de haut niveau consiste sur le calcul de similarité entre deux types d'alertes différentes à l'aide des similitudes métriques cosinus.

Les approches à base de similarité sont efficaces à cause de l'utilisation de clustering qu'il permet d'apprendre des données d'audit sans exiger à l'administrateur du système de fournir des descriptions explicites des différentes classes d'attaque. Cependant, les approches de similarité sont limitées dans leur capacité à découvrir des attaques coordonnées en raison du manque d'expressivité des alertes. Pour cela, beaucoup de chercheurs ont orienté leur recherches vers un autre aspect qui est la reconstruction des scénarios d'attaque en s'appuyant sur des scénarios d'attaques prédéfinies.

2.4.2 Corrélation d'alerte à base des scénarios d'attaques prédéfinies

Les systèmes de détection d'intrusions tentent de détecter les attaques qui se réalisent par une seule action. En revanche, les attaques deviennent de plus en plus compliquées, l'attaquant lance une chaîne d'actions (des actions élémentaires) du balayage de réseau jusqu'à le fait de réaliser son objectif de l'intrusion.

Cette problématique a motivé les chercheurs pour aller vers des approches qui visent à corréler les alertes en se focalisant sur les dépendances entre les alertes afin de construire des scénarios d'attaques.

Ces techniques prédisent que chaque attaque nécessite une séquence d'actions pour réussir, où ces actions visent à exploiter la vulnérabilité du système informatique. Dans cette technique, on construit des scénarios d'attaque en faisant correspondre des alertes à des modèles de scénarios prédéfinis fournis par des experts ou par apprentissage automatique.
En d'autre terme, chaque alerte de bas niveau est comparée aux étapes d'un certain nombre de scénarios d'attaque prédéfinis afin de les corréler avec le meilleur ajustement. Le principal défaut de cette technique est sa restriction aux scénarios d'attaques connues.
Une grande partie des approches de ce type reposent sur la définition des pré-conditions et des post-conditions des actions. F . Cuppen et al [CM02] proposent un modèle basique pour réduire les alertes dupliquées et pour corréler les alertes qui appartiennent au même scénario d'attaque en utilisant des fonctions de management des alertes, des fonctions de clustering et des fonctions pour fusionner les alertes. Ces fonctions sont représentées en logique du premier ordre.

Morin et al [MD03] proposent une approche qui consiste à effectuer une reconnaissance des modèles des scénarios d'attaques à base des séquences d'attaques prédéfinies. Ils appliquent le formalisme des chroniques pour agréger et corréler les alertes. Les chroniques sont des modèles temporels qui représentent des évolutions possibles du système observé. Une chronique est un ensemble d'événements, liés entre eux par des contraintes de temps, dont l'occurrence peut dépendre du contexte. Les informations temporelles disponibles permettent l'ordonnancement et la spécification des intervalles de temps entre deux occurrences d'événements.

Les chroniques sont utilisées dans de nombreuses régions distinctes [CD00]. Ils ont été principalement conçus pour analyser des séquences d'alarmes émises par des équipements dans des réseaux de télécommunications et des réseaux de distribution de tension. Ils sont également utilisés dans certaines sous-tâches d'un projet visant à représenter les flux des voitures dans le trafic routier. Dans le domaine médical, on cherche à dépister les symptômes de l'hépatite, à surveiller intelligemment les patients ou à détecter les arythmies cardiaques.

Les chroniques fournissent un Framework pour la modélisation des systèmes dynamiques. La reconnaissance des chroniques est basée sur un formalisme dans lequel le temps est fondamental.

Dans un tel système bien adapté à la détection d'intrusion, chaque alerte constitue un évènement daté et chaque attaque peut être décrite par une chronique.

Pour autant, les chroniques ont été initialement conçues pour la supervision des réseaux de télécommunications (détection de panne de composants réseaux). Les chroniques représentent alors des phénomènes de pannes. Elles sont écrites par des experts capables de décrire les manifestations d'une panne en matière de types d'événements et de délais. Les pannes sont des phénomènes récurrents et relativement déterministes. Un nombre relativement faible de chroniques permet donc de décrire la quasi-totalité des phénomènes de pannes. En revanche, dans le domaine de la détection d'intrusions, le déterminisme des scénarios employés par les attaquants est discutable, non seulement en terme d'alertes constatées, mais aussi en matière de délais inter attaques.

De manière générale, les chroniques peuvent à la fois représenter des phénomènes anormaux et des phénomènes normaux. En détection d'intrusions, il est donc envisageable d'utiliser les chroniques pour discriminer les faux positifs des vrais positifs en impliquant des événements anodins dans le processus de corrélation afin de consolider les alertes.

Les auteur dans [CGHO16]proposent un modèle de classification à base d'état pour détecter les attaques réalisées à partir d'un ensemble d'étapes en utilisant Hidden Markov Model afin d'identifier les étapes des attaques. Les auteurs adoptent cet technique de sort que la séquence des états d'attaque est un processus caché et est observé à travers une séquence d'observation émise.

La plupart de ces approches sont efficaces pour détecter les attaques connues, en revanche, elles sont incapables de détecter les nouvelles attaques. En plus, cette catégorie nécessite beaucoup de connaissances d'experts pour modéliser les étapes d'attaques, cette tâche est très chère et n'est pas toujours réaliste. De même, rater une étape peut amener à rater la détection d'un scénario plausible. Par exemple, certaines techniques utilisent des modèles ou des descriptions pour les scénarios d'attaques. Dans ce cas, la moindre différence entre l'attaque et le modèle peut prévenir la reconstruction du scénario.

2.4.3 Corrélation d'alerte à base de filtrage

Souvent, les attaques ciblent des services qui n'existent pas dans le système, par exemple une alerte spécifique sous Windows peut être non pertinente sur Linux. Ce qui conduit à générer un nombre énorme de faux positif. Plusieurs travaux visent à traiter les alertes en leur donnant une priorité selon leur impact sur le système à protéger. Cette priorité est donnée en se basant sur des algorithmes de filtrage spécifiques. Les auteurs dans [PFV02] proposent une technique de classement des alertes connue comme M-correlator,

prennent en considération la topologie et les objectifs opérationnels du système protégé. Cette approche classifie les alertes en se basant sur la probabilité que l'attaque réussisse et l'impact des alertes sur les ressources critiques du système protégé.

Qin et Lee [LQ05] calculent le score de priorité des alertes en se basant sur la sévérité de l'attaque et la pertinence des alertes sur le réseau et le système protégé. Les auteurs calculent les scores de priorité par des probabilités en se basant sur les réseaux Bayésiens, comme dans les réseaux Bayésiens, les données sont représentées par un graphe orienté, dans ce travail le noeud racine représente la priorité avec deux états "faible" et "élevé", les noeuds feuilles représentent l'intérêt avec une valeur de "faible", "moyen" ou "élevé", pour les autres noeuds feuilles qui sont système d'exploitation, application, ports/service et utilisateur le trois états sont "appariés", "non apparié" ou "inconnue".

Dans l'approche [KS12], chaque alerte est associée par un score initiale égale à zéro et l'utilisateur peut augmenter cette priorité ou la décrémenter, où le score est calculé en utilisant les attributs l'adresse IP source, l'adresse IP destination, port destination et la signature de l'alerte.

Dans l'approche [AAB12], les auteurs proposent un ensemble de métriques pour donner des scores et des priorités aux alertes afin de mieux réduire le nombre d'alertes. Ces métriques sont reliées à l'application de l'attaque contre la configuration du réseau qui vérifie si l'environnement satisfait la réalisation de l'attaque, l'état de configuration de l'IDS utilisé où l'importance de l'emplacement de l'IDS est relié par l'importance des machines et les services trouvés dans cet réseau, l'importance de la cible où l'objectif de cette métrique est d'augmenter le score des alertes liées aux activités suspectes qui ciblent les composants critiques du système, et la relation entre l'alerte courante et l'alerte précédente où le score d'une alerte est augmenter si cette alerte a une forte relation avec d'autres alertes déjà stockée. . La précision de la détection de la corrélation d'alerte dépend de la précision de la description des critères qui peuvent être trouvés dans les algorithmes de filtrage. Par conséquent, il y a un compromis entre le coût élevé de l'algorithme de filtrage et la complexité de calcul correspondante. Cependant, ce compromis n'a pas été abordé dans les approches de filtrage existantes [ZLK10].

Un travail récent est proposé [RAA19] où les auteurs proposent des calculer des scores pour les alertes afin de déterminer les alertes pertinentes et améliorer la qualité des alertes en procédant des techniques de post-traitement de bas niveau et de haut niveau. Dans le bas niveau, le filtrage des alertes non pertinentes ayant moins d'importance se fait par des scores de priorité en fonction de mesures de qualité. Les mesures de qualité utilisés sont la précision, l'exactitude, la fiabilité, l'intégrité et la pertinence des alertes en se basant sur l'état de la destination, vulnérable au attaque survenue, les mis à jour des signatures d'attaque, l'histoire des alertes générées par un capteur particulier et l'identification du modèle de l'attaque.

2.4.4 Corrélation d'alerte multi-steps

Les approches de cette catégorie visent à corréler les alertes afin de détecter les attaques inconnues. Dans [NCR02] P. Ning et al., représentent chaque alerte par ses pré-requises actions et ses conséquences. Ces actions sont représentées par des prédicats de premier ordre. Les alertes sont corrélées lorsqu'une pré-condition d'une alerte apparaît dans les post-conditions d'une autre. Ce travail introduit la notion de hyper-alert pour représenter les pré-requis et les conséquences de chaque alerte. Une hyper-alerte de type T est un triplet *(fait, pre-requis, conséquence)*, où (1) *fait* est un ensemble de noms d'attributs, chacun avec un domaine de valeurs associé, (2) un *pre-requis* est une combinaison logique de prédicats dont les variables libres sont toutes en fait et (3) *conséquence* est un ensemble de prédicats tels que toutes les variables libres en conséquence sont en fait. La notion de graphe de corrélation hyper-alert est aussi introduite dans ce travail, cette notion reflète les stratégies de haut niveau ou bien les étapes logiques derrière cette séquence d'attaques.

Un processus automatisé est proposé dans [LHT18], ce processus génère des milliers des règles de corrélation. Un modèle probabiliste graphique [BKM08] est proposé pour identifier les actions élémentaires des attaques complexes et corréler leurs alertes. Le processus de corrélation est traité comme un problème de classement. À partir d'un ensemble d'actions observées et d'un ensemble d'objectifs d'intrusion, l'objectif de l'approche est de déterminer les objectifs les plus plausibles d'un intrus. Lorsque les observations ne favorisent aucun plan d'attaque, l'approche permet également de confirmer que le trafic est normal. Cette approche comprend trois étapes principales :

1. Prétraitement de l'observation des données : cette étape de prétraitement concerne l'historique des observations. Le résultat de cette étape est un ensemble de données formatées.
2. Construction de Bayes naïfs : dans cette étape, la distribution de probabilité conditionnelle de chaque variable dans les Bayes naïfs est calculée.
3. Prévision des objectifs d'intrusion : lors de cette étape, les objectifs d'intrusion en appliquant le mécanisme d'inférence de Bayes naïfs sont prédits.

L'approche proposée dans [ALJ08], vise à améliorer la précision des systèmes de détection d'intrusions en utilisant un modèle théorique pour automatiser le raisonnement des alertes des différents senseurs.

Les auteurs dans [Als16] ont proposé un Framework pour découvrir la relation de causalité entre les alertes élémentaires qui appartiennent au même scénario d'attaque et réduire les alertes redondantes. Dans ce travail, le processus de corrélation est basé sur une extension des propriétés du modèle requiert fourni, le modèle proposé supprime les duplications en se basant sur un modèle graphique où les noeuds représentent les alertes à agréger et les axes représentent les relations de causalité entre les alertes.

La plupart de ces approches de corrélation d'alerte dans cette catégorie sont efficaces pour détecter les relations de causalité entre les alertes afin de détecter les attaques inconnues. Cependant, dans le processus de corrélation d'alerte, elles peuvent rater d'autres alertes.

En réalité, ces différentes approches reposent sur les informations trouvées dans les alertes, ce qui n'est pas suffisant pour améliorer le taux de détection et réduire le nombre de faux positifs. Pour cela, les chercheurs ont choisi d'enrichir la sémantique des alertes par l'exploitation du contexte des évènements.

Selon la définition fourni dans [ADB+99], le contexte est toute information qui peut décrire la situation des utilisateurs ou des objets. Dans le domaine de la détection d'intrusions on peut adapter cette définition comme toutes informations qui peut donner un aperçu sur la situation des machines et de réseau quand l'attaque a eu lieu. Comme un affect, le contexte peut enrichir la sémantique des alertes pour donner une interprétation significative aux alertes.

: L'approche proposée dans [SS14] vise à filtrer les alertes en utilisant le contexte et des similarités sémantique. Cette approche extrait un ensemble d'attributs des alertes afin de découvrir des similarités sémantiques entre ces alertes. La similarité sémantique concerne la similarité entre deux concepts de l'ontologie qui représente les données d'intrusions, où les concepts similaires sont structurés sous format d'un arbre de concepts. La similarité entre deux concepts dans une ontologie dépend des points communs entre ces concepts qui sont représentés par leurs relations avec leur plus petit ancêtre commun de l'ontologie, ainsi la différence entre les deux concepts qui est basé sur leurs emplacements dans l'ontologie.

La notion de l'arbre de concepts est utilisée afin de calculer la similarité sémantique entre les valeurs d'attributs symboliques des deux alertes a_i et a_j représentées par le vecteur d'attributs $\{x_i1, ...x_ip\}$ et $\{x_j1, ...x_jp\}$ respectivement. Cette métrique est calculée par l'équation suivante :

$$Sim(x_{ik}, x_{jk}) = 1 - \frac{path(x_{ik}, x_{jk}) + path(x_{jk}, x_{ik})}{depth(x_{ik}) + depth(x_{jk})}$$

Où $path(x_{ik}, x_{jk})$ est la longueur la plus courte du concept x_{ik} à l'ancêtre commun de x_{ik} et x_{jk} dans l'arbre de concepts et $depth(x_{ik})$ est la profondeur du concept x_{ik} dans l'arbre de concepts. Le résultat du calcule de cette similarité est une valeur entre o et 1 où la valeur 1 représente une correspondance totale et la valeur 0 représente la non similarité entre les concepts. Deux propriétés sont reliées à cette métrique, la première concerne les concepts généraux où la similarité sémantique entre les concepts généraux est inférieur par rapport la similarité sémantique entre les concepts spécialisés, la deuxième propriété concerne la similarité sémantique entre un concept est chaque ancêtre de ce concept est supérieure par rapport la similarité entre cette ancêtre et n'importe autre ancêtre du même

parent.

Les auteurs dans [MMDD09], ont prouvé que l'utilisation des alertes comme la seule ressource d'information n'est pas suffisant pour améliorer le taux de détection. Pour cela, un modèle de corrélation à base des informations contextuelles (la topologie, la cartographie du réseau et des informations sur la vulnérabilité) est proposé. Ce modèle est nommé M4D4 qui représente une version améliorée du modèle M2D2 [**?**].

L'objectif de M4D4 est de donner une représentation formelle des différentes sources d'informations intervenantes dans la détection d'intrusions telles que : le système surveillé, les vulnérabilités, les outils de sécurité et les événements observés. Dans ce modèle, un ensemble de règles en logique du premier ordre est proposées pour améliorer la qualité des alertes et réduire le nombre de fausses positives.

Les auteurs dans [SS14] ont proposé une approche qui vise à filtrer les alertes en se basant sur des informations contextuelles et des similarités sémantiques. Cette approche est basée sur l'extraction d'un ensemble d'attributs des alertes afin de découvrir les alertes similaires sémantiquement.

Gagnon et al. [GME09] ont utilisé la configuration de la cible (type de système d'exploitation et les applications) comme information contextuelle sous la supposition que la configuration de la machine cible a un impact sur le taux de détection des IDS.

Afin de fournir une représentation des connaissances puissantes en format unifié, plusieurs travaux ont été proposés [CDER09] [ICLC09] [UJP03] [VB$^+$10] [WG09] [LT10]. L. Coppolino et al. [CDER09] ont proposé une approche nommée Intrusion Detection and Diagnosis System (ID^2S) technologie qui vise à identifier les causes des attaques et estimer avec précision leurs conséquences sur les composants du système représentés par des ontologies. Les auteurs dans [ICLC09], ont proposé une approche hybride avec l'utilisation d'une analyse sémantique et une ontologie d'intrusion pour reconstruire les scénarios attaques connues et inconnues. Un autre travail intéressant [SFLZ14], propose un Framework de corrélation d'alertes, flexible et à base de contexte représenté par des ontologies. Cette approche consiste à intégrer les informations qui viennent à partir des capteurs de contexte, les bases de données de vulnérabilités connues et les bases de données d'attaques. Ainsi, elle consiste à corréler les informations existantes dans les ontologies, qui se fait via le moteur de corrélation en utilisant des règles logiques d'ontologie écrites en langage de règles Web Sémantique Query-Enhance (SQWRL).

Le tableau 2.1 représente certains critères prises pour comparer les travaux mentionnés dans cette étude. 1) Dynamique, ce qui signifie que le travail est basé sur un modèle ou une méthode dynamique, 2) Utilisation de la représentation des connaissances par défaut et d'exception, 3) Utilisation des informations trouvées dans les alertes, 4) Utiliser les préférences de l'administrateur, ce qui signifie utiliser l'expérience de l'administrateur, dans notre cas, nous traitons cette information comme une information contextuelle. 5)

Contexte, ce qui signifie utilisation des informations contextuelles pour fournir un raisonnement plus correct pour détecter les faux alertes, 6) Changement de contexte, ce qui signifie le traitement de l'arrivage de nouvelles connaissances sans refaire l'inférence du le début.

Approche	Description	Relation entre les alertes	Préférence	Contexte	Changement de contexte
Valdes et al [VS01]	Utiliser la similarité entre les alertes en utilisant un modèle probabiliste pour regrouper les alertes	✓	x	x	x
Xu et al. [XTT05]	Les auteurs utilisent des heuristiques pour calculer la similarité entre les attributs catégoriques affinés	✓	x	x	x
Hostiadi et al. [HSH19]	Deux niveaux d'analyse afin de découvrir la relation entre les alertes sans avoir des connaissances préalables	✓	x	x	x
F.cuppen et al [CM02]	Réduire les alertes redondantes en corrélant les alertes qui appartiennent à la même attaque en utilisant une logique de premier ordre	✓	x	x	x
Chen et al [CGHO16]	Adopter un model Hidden Markov afin de classifier les séquences des états d'attaque	✓	x	✓	x
Morin et al. [MD03]	Appliquer le formalisme des chroniques afin d'effectuer une reconnaissance des modèles d'attaques prédéfinis	x	✓x	✓x	✓x
Khalid Alsubhi et al[AAB12]	Utiliser la logique floue pour générer un score de priorité	✓	x	✓	x
Qin et al. [LQ05]	calculent le score de priorité des alertes en se basant sur la sévérité de l'attaque et la pertinence des alertes sur le réseau et le système protégé.	x	x	x	x
Sebastian Klüft et al [KS12]	Générer un classement en utilisant le vote de l'administrateur	✓	✓	x	x
Riyad et al.[RAA19]	Calculer des scores de priorités pour les alertes e fonction des techniques de mesures de qualité	x	x	✓	x
Peng et al [NCR02]	utiliser la relation de causalité entre les alertes utilisant des prédicats dans la logique du premier ordre	✓	x	✓	x
alserhani [Als16]	Découvrir les relations entre les alertes en se basant sur un modèle graphique	✓	x	✓	x
S.benferhat et al [BKM08]	identifier les actions élémentaires des attaques complexes et corréler leurs alertes.	✓	x	x	x
Benjamin Morin et al[MMDD09]	Définir un ensemble de règles sur la logique du premier ordre	x	x	✓	x
François Gagnon et al [GME09]	Montrer l'efficacité de l'utilisation de la configuration cible pour identifier les alertes critiques	x	x	✓	x
Alireza Sadighian et al[SFLZ14]	Utilise le contexte pour filtrer les alertes non pertinentes	✓	✓	✓	x

TABLE 2.1 – tableau comparatif des différentes approche

En comparant notre approche avec celles mentionnées dans cette étude, notre approche vise à identifier les alertes pertinentes. Ce fait peut aider l'opérateur de sécurité à analyser en premier temps les alertes qui s'adaptent à son intérêt. En plus, la plupart des approches dans la littérature ne prennent pas en considération l'aspect de changement de contexte qui peut se produire fréquemment. Ce changement se produit à cause de diverses raisons telles que la mise à jour de la configuration du système ou bien résoudre les failles des services et applications. Ces alertes sont pertinentes si la machine cible satisfait le contexte nécessaire pour qu'elle soit affectée par la vulnérabilité exploitée, mais ces alertes ne sont pas, dans

le cas de changement de contexte où la machine cible ne satisfait pas l'ancien contexte, même si ces alertes sont générées par la même attaque et pour la même cible.

Afin d'appliquer les règles et les concepts proposés en garantissant un raisonnement non monotone, nous avons utilisé la logique de description $JClassic_{\delta\epsilon}$ développé par Boustia et al. [BM12], cette logique fournit un algorithme traitable pour le calcul de la relation de subsomption entre les concepts et l'héritage des propriétés [BM10]. À la comparaison avec les autres formalismes non monotones [QR92] [PN93] [BH95] [CF97] [DTEK09], la logique de description $JClassic_{\delta\epsilon}$ est plus expressive avec une complexité polynomiale et utilisable dans un cadre pratique. Cette forme de logique de description fournit une nouvelle représentation des connaissances en incluant dans la définition des concepts les connaissances par défaut et exception. Autrement dit, chaque concept est défini par toutes ses propriétés, ce qui permet de captu rer les caractéristiques du contexte. Par conséquent, l'approche proposée correspond aux préférences de l'opérateur de sécurité.

2.5 Conclusion

Nous nous sommes focalisés dans ce chapitre sur les approches de la corrélation d'alerte existantes. Nous avons présenté une étude critique sur ces approches dont l'objectif principal de ces travaux est de répondre à la limite majeure des IDS par la réduction du nombre d'alertes générées particulièrement les fausses alertes.

Nous avons vu que dans la littérature, on distingue différentes catégories d'approches de corrélation d'alerte. Certaines utilisent des mesures de similarité entre les attributs d'alertes. Ces mesures sont considérés comme une méthode efficace pour regrouper les alertes, mais ils ne fournissent aucune information sur la relation certaine entre les attaques. D'autres visent à découvrir les scénarios d'attaques connues, cependant, ce type d'approche nécessite beaucoup de connaissances pour identifier et définir préalablement les étapes des attaques. Une autre catégorie d'approche donne une priorité aux alertes corrélées. Certaines approches visent à détecter les attaques inconnues de telle sorte à découvrir les relations logiques entre les alertes pour pouvoir reconstruire des plans d'attaque. En revanche, pour améliorer le taux de détection, la coopération des informations contextuelles a prouvé son avantage. Dans ce chapitre, nous avons discuté certaines approches qui décrivent la représentation de contexte comme information indispensable dans la détection d'intrusions, telles que, la topologie, la cartographie des systèmes surveillés, des informations sur les détecteurs, les vulnérabilités, et les attaques survenues.

À la lumière de cette étude, nous avons constaté que la plupart de ces approches de corrélation d'alertes révèlent certaines limitations telles que la nécessité de modéliser le changement de contexte et de fournir un raisonnement non-monotone qui répond aux besoins de l'opérateur de sécurité. Ces limitations représentent la motivation de la contribution représentée dans cette thèse.

Afin d'appliquer notre contribution, nous avons utilisé la logique de description $Jclassic_{\delta\epsilon}$ pour modéliser notre problème. Dans le chapitre qui suit, nous allons présenter cette logique en abordant sa sémantique et décrire en détail les algorithmes d'inférences qui permettent de calculer la subsomption et l'héritage des connaissances.

Chapitre 3

La logique de description non monotone JCLASSIC$_{\delta\epsilon}$ et les ontologies

3.1 Introduction

De nos jours, le raisonnement non-monotone est considéré comme une partie essentielle des approches logiques pour l'intelligence artificielle. La non-monotonie est apparue avec les tentatives de la représentation d'un raisonnement sur les actions et leur changement. Un raisonnement non-monotone peut être décrit comme la théorie de la création et de la sélection des hypothèses de manière dirigée.

Les hypothèses peuvent être abandonnées quand on apprend de nouvelles connaissances sur les circonstances qui provoquent une contradiction avec elles. Cependant, le raisonnement non monotone attribue un statut spécial de ces hypothèses, il les considère comme des hypothèses par défaut. Les hypothèses par défaut sont vues comme acceptables dans toutes les circonstances jusqu'à ce qu'elles soient en conflit avec d'autres défauts et le contexte actuel. En ce point, on peut noter que ce type de raisonnement est en conflit avec le raisonnement monotone. La monotonicité est une propriété qui caractéristique des inférences déductives découlant de la notion même d'une preuve : si "C" est provoqué d'un ensemble de propositions a, il est également provoqué d'un ensemble plus large $a \cup \{A\}$ [BDP97]. Le raisonnement non monotone est non monotonique dans ce sens, parce que l'ajout de nouveaux faits peut rendre invalide certaines hypothèses valides qui ont été faites avant.

Le développement d'un formalisme qui supporte un raisonnement non monotone, était l'objectif de plusieurs chercheurs. Les premiers travaux dans ce sens ont été ceux de Quantz and Royer [QR92] ; Padgham et Nebel [PN93] ; Padgham et Zhang [PZ93] ; Baader et Hollunder [BH95].

Ces travaux ont pour but d'étendre les DL en y incluant une forme limitée de raisonnement par défaut. Les défauts ne sont pas autorisés dans la définition même du concept. Les concepts ne sont définis que par leurs propriétés strictes alors que les connaissances défauts sont représentées par des règles incidentes. Le problème de ces solutions est que rares sont

les concepts qui peuvent être définis que par des propriétés strictes. Par conséquent, les classifieurs ne peuvent pas donner une classification complète des concepts, vu qu'ils sont partiellement définis. En outre, même si la combinaison des concepts partiellement définis et les règles sont suffisantes pour classifier les instances, il n'est pas clair comment ces approches prennent en considération l'exception de l'exception.

Une autre alternative intéressante a été proposée [DTEK09] qui utilise la logique des défauts de Reiter [Rei80] dans la définition de la base de connaissances en DL. Les auteurs commencent par faire une transformation de la théorie des défauts vers des programmes de requêtes conjonctives du premier ordre, ensuite, ils exploitent les contraintes additionnelles pour faire des recherches via des relations entre des conclusions par défaut et des justifications. Cette transformation correspond à l'opérateur d'exception (ϵ), un raffinement d'objet qui consiste à augmenter la description de l'objet par l'exception à travers d'autres concepts dont l'objet est une instance. Cette approche ne permet non plus de prendre en compte l'exception de l'exception.

Coupey and Fouqueré [CF97] ont introduit deux nouveaux connecteurs (δ) et (ϵ) qui permettent de représenter des connaissances "par défaut" et de type "exception", ce qui a donné la naissance d'une nouvelle logique de description nommée $AL_{\delta}\epsilon$, cette logique permet d'introduire les notions de défauts et d'exceptions dans la définition des concepts (i.e. dans la TBox), ce qui n'est pas disponible dans autres approches.

En effet, être capable de décrire les concepts avec les propriétés défauts, rend possible la définition des concepts qui ne peuvent pas être définis en utilisant que des propriétés strictes. Cependant, la logique de description $AL_{\delta}\epsilon$ ne peut pas être utilisée dans un cadre pratique à cause de son faible pouvoir d'expressivité. Ainsi, cette logique n'a pas été équipée avec des algorithmes d'inférence pour faire évoluer la base de connaissances.

Pour répondre à ces exigences, les auteurs dans [BM12] ont développé la logique de description non monotone $JClassic_{\delta\epsilon}$ afin de construire un modèle de contrôle d'accès dynamique et contextuel. Ce type de formalisme est inspiré de la logique $AL_{\delta}\epsilon$. Les auteurs ont étendu ce DL dans le but de maintenir une complexité polynomiale pour le calcul de l'héritage. Ainsi, les auteurs ont choisi d'ajouter les connecteurs de la logique *C-Classic* [BMPS+91] qui est une logique de description dotée avec une puissance d'expression très forte.

Au cours de ce chapitre, nous allons présenter la logique $Jclassic_{\delta\epsilon}$ en abordant sa sémantique et décrire en détail les algorithmes d'inférences qui permettent de calculer la subsumption et l'héritage des connaissances.

3.2 Langue terminologique $Jclassic_{\delta\epsilon}$ [BM10]

$Jclassic_{\delta\epsilon}$ se compose de l'ensemble de connecteurs de *C-Classic* avec la combinaison des connecteurs δ et ϵ. Le langage de description de $Jclassic_{\delta\epsilon}$ est défini en utilisant un

ensemble de **R** rôles primitifs, d'un ensemble **P** de concepts primitifs, des constants $\top$ et $\bot$, d'un ensemble **I** d'individus que l'on appelle *'individus classiques'*, et de la règle syntaxique suivante (C et D sont des concepts, P est un concept primitif, R est un rôle primitif, u est un nombre réel, n est un nombre entier et I_i des *'individus classiques'*) (voir Tableau 3.1) :

C,D→	$\top$	Le concept le plus général
	$\mid \bot$	Le concept le plus spécifique
	$\mid P$	Concept primitif
	$\mid C \sqcap D$	Conjonction de concepts
	$\mid \neg P$	Négation de concept primitif
	$\mid \forall r$: C	Restriction sur tous les rôles r
	$\mid Min_u$	ensemble des réels $> u$ (u est un nombre réel)
	$\mid Max_u$	ensemble des réels $\leq u$
	$\mid$ ONE-OF$\{I_1, .., I_n\}$	Concept en extension ($\{I_1, .., I_n\}$ sont des individus)
	$\mid$ r FILLS $\{I_1, .., I_n\}$	Sous-ensemble de valeur pour r
	$\mid$ r AT-LEAST n	Restriction minimum de nombre pour r (n est un entier)
	$\mid$ r AT-MOST n	Restriction maximum de nombre pour r
	$\mid \delta$ C	Concept par défaut
	$\mid C^{\epsilon}$	Exception sur le concept

TABLE 3.1 – La synatxe de $Jclassic_{\delta\epsilon}$

3.3 Cadre algèbrique pour $Jclassic_{\delta\epsilon}$

$Jclassic_{\delta\epsilon}$ est dotée d'un cadre algébrique ce qui permet de :

— Caractériser et de formaliser les différents types de subsomptions (descriptive et structurelle) dans $Jclassic_{\delta\epsilon}$.

— Prouver la correction et la complétude des algorithmes d'inférence.

— Avoir une adéquation entre la sémantique du système et les algorithmes d'inférence.

— Avoir des formes normales directement utilisables.

Ce cadre recouvre les différents aspects formels des notions de définition de concepts et de subsomption dans le langaget permet de doter $Jclassic_{\delta\epsilon}$ d'une sémantique intentionnelle (appelée $CL_{\delta\epsilon}$). Le calcul de dénotation de concepts dans $CL_{\delta\epsilon}$ est utilisé pour le calcul de subsomption dans l'algorithme $Sub_{\delta\epsilon}$. $CL_{\delta\epsilon}$ permet d'une part de montrer que $Sub_{\delta\epsilon}$ est correct et complet et d'autre part de donner un caractère formel du calcul de subsomption utilisé dans l'implémentation de $Jclassic_{\delta\epsilon}$.

Dans cette logique, la subsomption est considérée en point de vue descriptif et structurel.

3.3.1 Point de vue descriptif

Le calcul de subsomption se base sur un système équationnel appelé EQ, ce système est utilisé pour comparer les termes, il fournit une définition formelle pour les principales propriétés des connecteurs de $Jclassic_{\delta\epsilon}$ et il détermine les classes équivalentes des termes. Les propriétés définies par EQ sont représentées dans le tableau 3.2.
$\forall A, B, C \in JClassic_{\delta}\epsilon, I_j \in I, E_i \in 2^I$

TABLE 3.2 – Les propriétés des connecteurs de $Jclassic_{\delta\epsilon}$.

	axiome	la conjonction de concept est :
01	(A⊓ B) ⊓ C = A ⊓ (B ⊓ C)	associative
02	A⊓ B= B⊓ A	commutative
03	A⊓ A= A	idempotente
04	$\top \sqcap$ A= A	le concept le plus général dans la hiérarchie top $(\top)$ est l'élément neutre de la conjonction
05	$\bot \sqcap$ A= $\bot$	le concept le plus spécifique bottom $(\bot)$ est un élément absorbant
06	($\forall$ R : A) ⊓ ($\forall$ R : B) =$\forall$ R(A ⊓ B)	le connecteur R : C est distributif par rapport à la conjonction de concepts.
07	$\forall$ R : $\top = \top$	Il y a équivalence entre certains termes de $Jclassic_{\delta\epsilon}$ et le top $(\top)$, ces termes représentent de fausses restrictions sur des rôles
08	ONE-OF E_1⊓ ONE-OF E_2 = $ONE-OF(E_1 \cap E_2)$	ONE-OF $E_1 subsume ONE-OF E_2 si E_1 \subseteq E_2$
09	MIN m ⊓ MIN n = MIN maxi(m,n)	MIN m subsume n si n $\geq$ m
10	MAX m ⊓ MAX n = MAX mini(m,n)	MIN m subsume MAX n si n $\leq$ m
11	R FILLS E_1⊓ R FILLS E_2 = R FILLS ($E_1 \cup E_2$)	R FILLS $E_1 subsume RFILLS E_2 si E1 \subseteq$ E2
12	R FILLS $\emptyset = \top$	Il y a équivalence entre certains termes de $Jclassic_{\delta\epsilon}$ et le top $(\top)$, ces termes représentent de fausses restrictions sur des rôles
13	R AT-LEAST m ⊓ R AT-LEAST n = R AT-LEAST maxi(m,n)	R AT-LEAST m subsume R AT-LEAST n si n $\geq$ m
14	R AT-LEAST 0=$\top$	Il y a équivalence entre certains termes de $Jclassic_{\delta\epsilon}$ et le top $(\top)$, ces termes représentent

		de fausses restrictions sur des rôles
15	R AT-MOST m ⊓ R AT-MOST n = R AT-MOST mini(m,n)	R AT-MOST m subsume R AT-MOST n si n ≤ m
16	R ⊓AT-MOST ⊓0= ∀ R : ⊥	Il y a équivalence entre AT-MOST 0 et ∀ : ⊤. Ces deux termes représentent l'absence de fillers du rôle R
17	R ⊓FILLS $\{I_1...I_n\}$= R ⊓FILLS $\{I_1...I_n\}$ ⊓ R ⊓ AT-LEAST ⊓n	R ⊓AT-LEAST ⊓n ⊓subsume ⊓R ⊓FILLS $\{I_1...I_n\}$
18	∀ R : ⊓ONE-OF $\{I_1...I_n\}$ = ∀ R : ⊓ONE-OF $\{I_1...I_n\}$ ⊓ R ⊓AT-MOST ⊓n	R ⊓AT-MOST ⊓n ⊓subsume ∀ R : ⊓ONE-OF $\{I_1...I_n\}$
19	R ⊓AT-LEAST ⊓n ⊓ ∀R : ⊓ONE-OF $\{I_1...I_n\}$ = R ⊓AT-LEAST ⊓n ⊓ ∀ R : ONE-OF $\{I_1...I_n\}$ ⊓ R ⊓ FILLS $\{I_1...I_n\}$	R ⊓AT-LEAST ⊓n ⊓ ∀R : ⊓ ONEOF $\{I_1...I_n\}$ est équivalente à R ⊓ AT-MOST ⊓n ⊓ R ⊓ FILLS $\{I_1...I_n\}$
20	R ⊓ AT-MOST ⊓n ⊓ R ⊓ FILLS $\{I_1...I_n\}$ = R AT-MOST n ⊓ R FILLS $\{I_1...I_n\}$⊓ ∀R : ⊓ONE-OF $\{I_1...I_n\}$	R ⊓ AT-LEAST ⊓n ⊓∀ R : ⊓ ONEOF $\{I_1...I_n\}$ est équivalente à R AT-MOST n ⊓R ⊓ FILLS $\{I_1...I_n\}$
21	ONE-OF ∅	ONE-OF ∅
22	MIN ⊓m ⊓MAX ⊓n (if ⊓n≤m)	MIN ⊓m ⊓MAX ⊓n (if ⊓n≤ m)
23	R ⊓ AT-LEAST ⊓m ⊓∀ R ⊓AT-MOST ⊓n (if ⊓n < m)	R ⊓AT-LEAST ⊓m ⊓∀R ⊓AT-MOST ⊓n (if ⊓n < m)
24	R ⊓FILLS ⊓E_1⊓∀ : ONE-OF ⊓E_2(*if*⊓ E_2 E_1)	R ⊓ FILLS E_1⊓∀ : ONE-OF ⊓ E_2(*if*⊓ E_2 E_1)
25	$(\delta A)^\epsilon = A^\epsilon$	une exception n'a de sens que si elle porte sur un concept par défaut
26	$\delta(A \sqcap B) = (\delta A) \sqcap (\delta B)$	reflète la distributivité du connecteur δ par rapport à la conjonction de concepts
27	A⊓δ A= A	A est subsumé par δA.
28	$A^\epsilon \sqcap \delta A = A^\epsilon$	A^ϵ est subsumé par δA

29	$\delta\delta A = \delta A$	exprime la propriété d'idempotence du connecteur δ

Les vingt-quatre premiers axiomes correspondent aux propriétés des connecteurs de *C-Classic* [BMPS+91]. Les propriétés des connecteurs de δ et ϵ définies en $AL_{\delta\epsilon}$ [CF97] sont représentées par les axiomes de 25 à 29.

Les propriétés 8, 9, 10, 11, 13, 15 représentées dans le tableau 3.2 soulignent des relations de subsomption entre des connecteurs identiques.

La subsomption descriptive : la subsomption descriptive est représentée par la relation $\sqsubseteq_d$ qui est une relation d'ordre partiel permettant d'ordonner les termes de $Jclassic_{\delta\epsilon}$. L'égalité (modulo les axiomes d'EQ) entre deux termes de $Jclassic_{\delta\epsilon}$ notée $=_{EQ}$.$=_{EQ}$ est une relation de congruence qui partitionne l'ensemble des termes de $Jclassic_{\delta\epsilon}$. I.e. $=_{EQ}$ permet de former des classes d'équivalence entre les termes.

La subsomption descriptive est définie à l'aide de cette relation de congruence et de la conjonction de concept de la manière suivante :
Soient C, D deux termes de $Jclassic_{\delta\epsilon}$, $C \sqcap D$, i.e. D subsume descriptivement C, ssi $C \sqcap D =_{EQ} C$.

Un point de vue structurel est utilisé pour le calcul de la subsomption pour pouvoir manipuler les termes en $Jclassic_{\delta\epsilon}$. Ce point de vue structurel permet d'une part de formaliser le calcul de la subsomption et d'autre part de doter $Jclassic_{\delta\epsilon}$ d'une sémantique intentionnelle.

3.3.2 Point de vue structurel du cadre algébrique

Afin de pouvoir calculer la subsomption dans $Jclassic_{\delta\epsilon}$, $CL_{\delta\epsilon}$ est défini et qui correspond à une sémantique intentionnelle dans laquelle les concepts sont représentés par une forme normale de leur ensemble de propriétés.

Le calcul de subsomption du point de vue structurel consiste à comparer les formes normales calculées en appliquant un homomorphisme de l'ensemble des termes de $Jclassic_{\delta\epsilon}$ vers les éléments de $CL_{\delta\epsilon}$

3.4 $CL_{\delta\epsilon}$: une sémantique intentionnelle pour $Jclassic_{\delta\epsilon}$

Les éléments de $CL_{\delta\epsilon}$ sont des représentations intentionnelles canoniques des termes de $Jclassic_{\delta\epsilon}$ (i.e. des formes normales de leurs ensembles de propriétés), on appelle les éléments de $CL_{\delta\epsilon}$ formes normales ou dénotations. La définition de $CL_{\delta\epsilon}$ passe par la définition d'un *homomorphisme* h de l'ensemble des termes de $Jclassic_{\delta\epsilon}$.

3.4.1 Structure des éléments

Les éléments de $CL_{\delta\epsilon}$ sont formés d'une paire de sextuples. Les deux sextuples ont une structure identique ; le premier permet de représenter les propriétés strictes et le deuxième permet de représenter les propriétés par défaut. Les cinq premiers champs correspondent aux propriétés de *C-Classic*. Le sixième champ est un champ permettant de représenter les exceptions. Le champ exception comprend un ensemble éventuellement vide de sextuples où chaque sextuple représente un concept excepté (on rappelle qu'une exception n'a pas de sens que si elle est portée sur une propriété défaut.). D'un point de vue intuitif la structure des éléments de $CL_{\delta\epsilon}$ est la suivante :

Définition 3.4.13 *Un élément de $CL_{\delta\epsilon}$ correspondant à un terme T de $Jclassic_{\delta\epsilon}$ est un couple $\langle t_\theta, t_\delta \rangle$ où t_θ contient les propriétés strictes de T et t_δ ses propriétés défauts. t_θ et $t_\delta$sont des sextuplets définis comme suit : (dom,min,max, π,r,ϵ) où :*
*- **dom** est soit un ensemble d'individus si la définition de T contient une propriété ONE-OF ou le symbole spécial UNIV sinon,*
*- **min** (resp. max) est soit un réel si $\top$ contient une propriété MIN (resp. MAX), ou le symbole spécial MIN-R (resp. MAX-R) sinon,*
- π est l'ensemble des concepts primitifs appartenant à $\top$,
*- **r** est un ensemble d'éléments définis comme suit : $\langle$ R ; fillers ; least ; most ; c$\rangle$*
où :
*- **R** : un nom de rôle.*
*- **Fillers** : ensemble d'individus, si $\top$ contient une propriété R Fills, ou θ sinon,*
*- **least (resp. most)** : est un entier, si $\top$ contient une propriété R ATLEAST (resp. R AT-MOST), else 0 (resp. NOLIMIT).*
*- **c** : est la forme normale de C, si $\top$ contient une propriété $\forall$ R : C.*
- Notation : La structure complète est notée :
($\langle$ ($t_{\theta dom}$; $t_{\theta max}$; $t_{\theta min}$; $t_{\theta\pi}$; t_θ r ; $t_{\theta\epsilon}$; ($t_{\theta dom}$; $t_{\delta max}$; $t_{\delta min}$; $t_{\delta\pi}$; $t_{\delta r}$; $t_{\delta\epsilon}\rangle$))
Example :
La forme normale du concept A $\equiv$ B $\sqcap$ C $\sqcap\delta$ D est :
fn (A) = ($\langle$ Univ, Min-R ; Max-R, {B, C} , 0, 0$\rangle$; $\langle$ Univ ; Min-R, Max-R ; {B, C, D} , 0, 0$\rangle$).

Définition 3.4.14 (Homomorphisme h) *L'interprétation des connecteurs et des constantes de $CL_{\delta\epsilon}$ est donnée dans le tableau 3.3. b0 est une constante utilisée comme dénotation de $\bot$.*

La subsomption structurelle qui s'appuie sur la comparaison d'éléments de $CL_{\delta\epsilon}$ est définie comme suit :
Deux termes C et D de $Jclassic_{\delta\epsilon}$ sont équivalents structurellement si et seulement si leurs

$Jclassic_{\delta\epsilon}$	$CL_{\delta\epsilon}$
$\top$	$\langle$*(UNIV, MIN-R, MAX-R,*$\emptyset, \emptyset, \emptyset$*),* (UNIV, MIN-R, MAX-R, $\emptyset$, $\emptyset$, $\emptyset$)$\rangle$
P	$\langle$(UNIV, MIN-R, MAX-R,P, $\emptyset$, $\emptyset$), (UNIV, MIN-R, MAX-R, $\emptyset$, $\emptyset$, $\emptyset$)$\rangle$
ONE-OF E	$\langle$(E, MIN-R, MAX-R,P, $\emptyset$, $\emptyset$), (E, MIN-R, MAX-R, $\emptyset$, $\emptyset$, $\emptyset$)$\rangle$
MIN u	$\langle$(UNIV, u, MAX-R,$\emptyset$, $\emptyset$, $\emptyset$), (UNIV, u, MAX-R, $\emptyset$, $\emptyset$, $\emptyset$)$\rangle$
$MAXu$	$\langle$(UNIV, MIN-R, u,$\emptyset$, $\emptyset$, $\emptyset$), (UNIV, MIN-R, u, $\emptyset$, $\emptyset$, $\emptyset$)$\rangle$
$\forall R : C(C\neg \equiv \top etC\neg \bot)$	$\langle$(UNIV, MIN-R, MAX-R,$\emptyset$, {$\langle$ R, $\emptyset$, 0, |c$_\theta$.dom|, c$\rangle$ },$\emptyset$), (UNIV, MIN-R, MAX-R, , {$\langle$ R,$\emptyset$,$\emptyset$, |c$_\theta$.dom|, c$\rangle$},$\emptyset$)$\rangle$
$\forall$ R :C et C $\equiv \bot$	$\langle$ (UNIV, MIN-R, MAX-R,$\emptyset$, {$\langle$ R, $\emptyset$, 0, 0, b$_0\rangle$},$\emptyset$ (UNIV, MIN-R, MAX-R,$\emptyset$, { $\langle$ R,$\emptyset$, 0, 0, b$_0\rangle$}, $\emptyset$) $\rangle$
$\forall$ R : Cet C $\equiv \top$	$\langle$(UNIV, MIN-R, MAX-R,$\emptyset$, $\emptyset$, $\emptyset$), (UNIV, MIN-R, MAX-R, $\emptyset$, $\emptyset$, $\emptyset$)$\rangle$
R FILLS E(E $\neq \emptyset$)	$\langle$(UNIV, MIN-R, MAX-R,$\emptyset$, {$\langle$ R, E, |E|, NOLIMIT, t$\rangle$},$\emptyset$), (UNIV, MIN-R, MAX-R, $\emptyset$, {$\langle$ R, E, |E|, NOLIMIT, t$\rangle$},$\emptyset$) $\rangle$
R FILLS $\emptyset$	$\langle$(UNIV, MIN-R, MAX-R,$\emptyset$, $\emptyset$,$\emptyset$), (UNIV, MIN-R, MAX-R, $\emptyset$,$\emptyset$,$\emptyset$)$\rangle$
R AT -LEAST n(n$\neq$ 0)	$\langle$(UNIV, MIN-R, MAX-R,$\emptyset$, {$\langle$ R, $\emptyset$, n, NOLIMIT, t$\rangle$},$\emptyset$), (UNIV, MIN-R, MAX-R, $\emptyset$, {$\langle$ R, 0, n, NOLIMIT, t$\rangle$}, $\emptyset$)$\rangle$
R AT-LEAST 0	$\langle$(UNIV, MIN-R, MAX-R,$\emptyset$,$\emptyset$,$\emptyset$), (UNIV, MIN-R, MAX-R, $\emptyset$,$\emptyset$,$\emptyset$)$\rangle$
R AT-LEAST n(n$>$ 0)	$\langle$(UNIV, MIN-R, MAX-R,$\emptyset$, {$\langle$ R, $\emptyset$, 0, n, t$\rangle$}, $\emptyset$), (UNIV, MIN-R, MAX-R, $\emptyset$, {$\langle$ R, $\emptyset$, 0, n, t$\rangle$}, $\emptyset$)$\rangle$
R AT-MOST 0	$\langle$(UNIV, MIN-R, MAX-R,$\emptyset$, {$\langle$ R, $\emptyset$, 0, 0, b$_0\rangle$}, $\emptyset$), (UNIV, MIN-R, MAX-R, $\emptyset$, {$\langle$ R,$\emptyset$,0, 0, b$_0\rangle$}, $\emptyset$) $\rangle$
$C \sqcap D$	C $\otimes$ d
δC	$\langle$(UNIV, MIN-R, MAX-R,$\emptyset$,$\emptyset$,$\emptyset$), c$_\delta$)$\rangle$
C^ϵ	$\langle$(UNIV, MIN-R, MAX-R,$\emptyset$, $\emptyset$, c_δ), ($c_\delta.dom, c_\delta.max, c_\delta.min, c_\delta\pi, c_\delta r, c_\delta\epsilon \cup c_\delta$)$\rangle$
$\bot$	b_0

TABLE 3.3 – Interprétation des connecteurs et des constantes de $Jclassic_{\delta\epsilon}$ en $CL_{\delta\epsilon}$.

formes normales sont égales.
La subsomption structurelle C $\sqsubseteq_S$ D est une relation d'ordre partiel. L'égalité structurelle de deux termes de $Jclassic_{\delta\epsilon}$ est notée $=_{CL\delta\epsilon}$. $=_{CL\delta\epsilon}$ est une relation de congruence comme l'equation $=_{EQ}$ dans le cadre de la subsomption descriptive.
La subsomption structurelle est définie à partir de la relation de congruence $=_{CL_{\delta\epsilon}}$ et de la conjonction de concepts comme suit :
Soient C, D deux termes de JClassic $JClassic_{\delta\epsilon}$, C $\sqsubseteq_S$ D, i.e. D subsume structurellement C, ssi C $\sqcap$ D $=_{CL\delta\epsilon}$ C
Il y a une équivalence entre la subsomption descriptive et la subsomption structurelle, formellement, soient C et D deux termes de $Jclassic_{\delta\epsilon}$, C $\sqsubseteq_S$ D $\Leftrightarrow$ C $\sqsubseteq_d$ D
L'algorithme de subsomption est représenté dans la section suivante .

3.5 Algorithme de calcul de subsomption $Sub_{\delta\epsilon}$ [Bou11]

L'algorithme de subsomption $Sub_{\delta\epsilon}$ s'appuie sur le calcul de la subsomption structurelle décrite dans la section précédente. $Sub_{\delta\epsilon}$ est un algorithme de calcul de subsomption du type *normalisation-comparaison* ; il est composé d'une étape de normalisation des descriptions suivie d'une étape de comparaison purement syntaxique des formes normales obtenues.
Étant donné deux termes C et D de $Jclassic_{\delta\epsilon}$, répondre à la question 'D subsume t-il C ?' nécessite l'application de la procédure suivante :
Les formes normales de C et 'C$\sqcap$D' sont calculées en appliquant une procédure de normalisation. Si les deux formes normales obtenues sont égales, l'algorithme retourne "Oui" sinon il retourne "Non" (Voir Algorithme 1)

Algorithm 1 $Sub_{\delta\epsilon}(C, D)$

Require: C and D two concepts of $Jclassic_{\delta\epsilon}$
Ensure: Response "Yes" or "No" to the question "D does it subsume C ?" //Calculation of normal forms.
FN(C) ← Normalisation(C)
FN(D) Normalisation(D)
//Bottom treatment(⊥).
if FN(C)= b_0 **then**
Response ← Yes FN(C ⊓ D) = b_0
Response ← No
else//Comparison of the normal forms obtained.
Compar(FN(C)$_\theta$, FN(C ⊓ D)$_\theta$,resp1)
if Resp1= "Yes" **then**
Compar(FN(C)$_\delta$, FN(C ⊓ D)$_\delta$,resp1)
Response ← resp1
elseResponse ← "No"
end if
end if

L'algorithme de subsomption Sub$_{\delta\epsilon}$ fait appel à une fonction de normalisation décrite dans l'algorithme 3, où on distingue deux termes de la forme ∀ R :C avec C ≠ ⊤ et ceux comportant le connecteur ⊓ que l'on appelle terme composé (voir Algorithme 5), d'autres termes, appelés termes simples (voir Algorithme 4).

La procédure de normalisation utilise les fonctions sémantiques union-uple et ⊗ qui calcule respectivement l'union de deux sextuplets (⊗) et de deux formes normales.

L'opération de l'union de deux éléments de $CL_{\delta\epsilon}$ (⊗) est définie comme suit :
Pour toute forme normale c de $CL_{\delta\epsilon}$ on a : b_0⊗ c =c(⊗) b_0.
Soient c et d deux formes normales de $CL_{\delta\epsilon}$ différentes de b_0 de la forme :
$c = \prec (c_\theta, c_\delta \succ$ *et* $d = \prec (d_\theta, d_\delta)\succ$
c ⊗ d = ≺(c$_\theta$, d$_\theta$) , (c$_\delta$, d$_\delta$)≻
Cette définition fait appel à une opération d'union de deux sextuplets de $CL_{\delta\epsilon}$ **(⊗)**. L'union de deux sextuplets se fait sur chacun de leurs champs *(dom, min, max,* π*, r,* ∈*)*.

Le calcul de l'union est donné sous la forme d'une fonction sémantique appelée union-uple(A, B) (pour A ⊗ B). La définition de cette fonction est donnée dans l'algorithme 2 [Bou11].

Algorithm 2 $union-uplet(C, D)$

Require: two sextuplets A and B : $(a.dom,\ a.min,\ a.max,\ a.\pi,\ a.r,\ a.\epsilon)$ *and* $(b.dom,\ b.min,\ b.max,\ b.\pi,\ b.r,\ b.\epsilon)$

Ensure: (u.dom, u.min, u.max, u.π, u.r, u.ϵ) the sextuplet result of the union of A and B

u$_{dom}$ ← c$_{dom}$ ∪d$_{dom}$
u$_{min}$ ← max(c$_{min}$ ∪ d$_{min}$)
u$_{max}$ ← min(c$_{max}$ ∪ d$_{max}$)
uπ ← c$_{\pi}$ ∪ d$_{\pi}$
u$_{\epsilon}$ ← c$_{\epsilon}$∪ d$_{\epsilon}$
u$_r$ ← ∅
for ⟨ r, fills1, least1, most1, c1 ⟩ ∈ a$_r$ **do**
  if ∃⟨ r, fills2, least2, most2, c2 ⟩ ∈ b$_r$ **then**
    u$_r$ ← u$_r$ ∪⟨ r, fills, least, most, c ⟩ with :
    C ← c1 ⊗ c2
    fills0 ← fills1 ∪ fills2
    least ← max(least1, least2,| fillers |)
    most ← min (most1, most2,| C$_{\theta}$. dom |)
  else
    u$_r$ ← u$_r$ ∪⟨ R, c1 ⟩
  end if
end for
for all (⟨ R, c ⟩ ∈ b$_r$), such as ∄ of name element R in a$_r$ **do**
  u$_r$ ← u$_r$ ∪⟨ R, c ⟩

end for

La procédure de la normalisation est représentée dans l'algorithme 3 [Bou11].

Algorithm 3 procédure de normalisation

Require: D is a description of a concept of $Jclassic_{\delta\epsilon}$

Ensure: of D in $CL_{\delta\epsilon}$ fn(D), D has the form $D \equiv T1 \sqcap T2 \sqcap ... \sqcap Tn$ with $n \geq 1$

if (D is ⊤) or (D is ⊥) or (D is a primitive concept) or (D is ONE-OF E) or (D is MIN u) or (D is MAX u) or (D is R FILLS E ($E \neq \emptyset$)) or (D is R FILLS ∅) or (D is R AT-LEAST n ($n \geq 1$)) or (D is R AT-LEAST 0) or (D is R AT-MOST n ($n \geq 1$)) or (D is R AT-MOST 0) or (D is $\forall R : \top$) or (D is $\forall R : \bot$) **then**
  termsimple
else
  termcomp
end if

Le traitement des termes simples et termes composés est décrit par les deux algorithmes 4 et 5 respectivement [Bou11].

Algorithm 4 Procédure de terme simple

if D is$\top$ **then**
FN(D) $\leftarrow \prec$ (UNIV, MIN-R, MAX-R, $\emptyset, \emptyset, \emptyset$),(UNIV, MIN-R, MAX-R, $\emptyset, \emptyset, \emptyset)\succ$
end if
if D is $\bot$ **then**
FN(D) $\leftarrow$ b_0
end if
if D is a primitive concept **then**
FN(D) $\leftarrow \prec$ (UNIV, MIN-R, MAX-R, { P }, $\emptyset$, $\emptyset$),(UNIV, MIN-R, MAX-R, $\{P\}, \emptyset, \emptyset)\succ$
end if
if D is ONE-OF E **then**
FN(D) $\leftarrow \prec$(E, MIN-R, MAX-R, $\emptyset, \emptyset, \emptyset$),(E, MIN-R, MAX-R, $\emptyset, \emptyset, \emptyset)\succ$
end if
if D is MIN u **then**
FN(D) $\leftarrow \prec$ (UNIV, u, MAX-R, $\emptyset, \emptyset, \emptyset$),(UNIV, u, MAX-R, $\emptyset, \emptyset, \emptyset)\succ$

end if
if D is MAX u **then**
FN(D) $\leftarrow \prec$(UNIV, MIN-R, u, $\emptyset, \emptyset, \emptyset$),(UNIV, MIN-R, u, $\emptyset, \emptyset, \emptyset)\succ$

end if
if D is R FILLS E (E$\neq \emptyset$) **then**
FN(D) $\leftarrow \prec$ (UNIV, MIN-R, MAX-R, $\emptyset$, {$\prec$ R, E, | E |, NOLIMIT, t $\succ$}, $\emptyset$),(UNIV, MIN-R, MAX-R, $\emptyset$, {$\prec$ R, E, $|E|$, NOLIMIT, t $\succ$}, $\emptyset)\succ$

end if
if (D is R FILLS $\emptyset$) ou (D est R AT-LEAST 0) ou (D est $\forall$ R :$\top$) **then**
FN(D) $\leftarrow$ t
end if
if D is R AT-LEAST n (n $\geq$ 1) **then**
FN(D) $\leftarrow \prec$ (UNIV, MIN-R, MAX-R, $\emptyset$, {$\prec$ R, $\emptyset$, n, NOLIMIT, t$\succ$}, $\emptyset$),(UNIV, MIN-R, MAX-R, $\emptyset$, {$\prec$ R, $\emptyset$, n, NOLIMIT, t $\succ$}, $\emptyset) \succ$

end if
if D is R AT-MOST n(n ≥ 1) **then**
FN(D) $\leftarrow \prec$ (UNIV, MIN-R, MAX-R, $\emptyset$, {$\prec$ R, $\emptyset$, 0,n, t $\succ$}, $\emptyset$),(UNIV, MIN-R, MAX-R, $\emptyset$, {$\prec$ R, $\emptyset$, 0,n, t$\succ$}, $\emptyset)\succ$

end if
if D is R AT-MOST 0 **then**
FN(D) $\leftarrow \prec$ (UNIV, MIN-R, MAX-R, $\emptyset$, {$\prec$ R, $\emptyset$, 0,0, $b_0\succ$}, $\emptyset$),(UNIV, MIN-R, MAX-R, $\emptyset$, {$\prec$ R, $\emptyset$, 0,0, $b_0\succ$}, $\emptyset) \succ$

end if
if D is $\forall$ R :$\bot$ **then**
FN(D) $\leftarrow \prec$ (UNIV, MIN-R, MAX-R, $\emptyset$, {$\prec$ R, $\emptyset$, 0,0, $b_0\succ$}, $\emptyset$),(UNIV, MIN-R, MAX-R, $\emptyset$, {$\prec$ R, $\emptyset$, 0,0, $b_0\succ$}, $\emptyset)\succ$

end if

Algorithm 5 Procédure de terme composé

if D is $\forall$ R : C **then**
FN(C) $\leftarrow$ Normalisation (C)
if FN(C) = t **then**
FN(D) = t
else
if FN(C)=b_0 **then** $FN(D) \leftarrow \prec$ (UNIV, MIN-R, MAX-R, $\emptyset$, {$\prec$R, $\emptyset$, 0,0, $b_0 \succ$}, $\emptyset$),(UNIV, MIN-R, MAX-R, $\emptyset$, {$\prec$ R, $\emptyset$, 0,0, $b_0 \succ$} $\emptyset$)$\succ$

FN(D) $\leftarrow \prec$ (UNIV, MIN-R, MAX-R, $\emptyset$, {$\prec$ R, $\emptyset$, 0,| $C_\theta.dom$ |, $C \succ$}, $\emptyset$),(UNIV, MIN-R, MAX-R, $\emptyset$, {$\prec$R, $\emptyset$, 0,| $C_\theta.dom$ |, C$\succ$}, $\emptyset$) $\succ$

if D is δ C **then**
FN(C) $\leftarrow$ Normalisation (C)

FN(D) $\leftarrow \prec$ (UNIV, MIN-R, MAX-R, $\emptyset$, $\emptyset$, $\emptyset$), FN(C_δ)$\succ$
end if
if D is C^ϵ **then**
FN(D)$_{\theta dom}$ $\leftarrow$ UNIV
FN(D)$_{\theta Min}$ $\leftarrow$ MIN-R
FN(D)$_{\theta Max}$ $\leftarrow$ MAX-R
FN(D)$_{\theta\pi}$ $\leftarrow \emptyset$
FN(D)$_{\theta r}$ $\leftarrow \emptyset$
FN(D)$_{\theta\epsilon}$ $\leftarrow$ Calcul-index(FN(C)$_\delta$)
FN(D)$_{\delta dom}$ $\leftarrow$ FN(C)$_{\delta dom}$
FN(D)$_{\delta min}$ $\leftarrow$ FN(C)$_{\delta min}$
FN(D)$_{\delta max}$ $\leftarrow$ FN(C)$_{\delta max}$
FN(D)$_{\delta\pi}$ $\leftarrow$ FN(C)$_{\delta\pi}$
FN(D)$_{\delta r}$ $\leftarrow$ FN(C)$_{\delta r}$
FN(D)$_{\delta\epsilon}$ $\leftarrow$ FN(C)$_{\delta\epsilon}$
end if
if D is A $\sqcap$ B **then**
FN(A) $\leftarrow$ Normalisation (A)
FN(A) $\leftarrow$Normalisation (B)
FN(D) $\leftarrow$ FN(A) $\otimes$ FN(B) //Union of two normal forms
end if
=0

Le calcul des champs ϵ de la forme normale fait appel à un codage permettant de représenter les concepts exceptés (ou plus précisément le sextuplet défaut de la forme normale des concepts) de manière plus concise. La table d'index comprenant des couples (nombre, sextuple) est définie dans l'algorithme 6 [Bou11] comme suit :

Algorithm 6 Procedure calcul-index(s) (s est un sextuplet)

if ∃ une paire (n,s) dans la table d'index **then**
  calcul-index(s) ← n
else//On crée un nouveau couple dans la table d'index (m,s) où m= nombre de couple dans la table d'index + 1
  calcul-index(s) ← m

end if=0

Remarque : le calcul d'index d'un sextuple renvoie un nombre correspondant au codage du sextuple qui sera utilisé dans la forme normale et permet en même temps de mettre à jour la table d'index.

Le gain de place obtenu est important dans le cas où la description comprenant une série d'exception (ϵ) imbriqués.

3.5.1 Procédure de comparaison des formes normales

La procédure *Compar* teste l'égalité entre deux sextuples t1 et t2. Où t1 (resp. t2) est de la forme : ($t_{i.dom}$, $t_{i.min}$, $t_{i.max}$, $t_{i\pi}$, $t_{i.r}$, $t_{i.\epsilon}$) avec i=1 (resp. i=2).
Compar fait appel à la procédure *Compar-roles* qui permet de tester l'égalité des ensembles dénotant les rôles. (voir algorithme 7 et 8) [Bou11]

Algorithm 7 Procédure Compar(t1,t2,Réponse)

if ($t1_{dom} \neq t2_{dom}$) or ($t1_{min} \neq t2_{min}$) or ($t1_{max} \neq t2_{max}$) or ($t1_{\pi} \neq t2_{\pi}$) or ($t1_{\epsilon} \neq t2_{\epsilon}$) **then**
  Response ← "No"
else
  //Role Field Comparison
  Compar-roles($t1_r$,$t2_r$,Resp)
  Response ← Resp
end if

Algorithm 8 Compar-roles(A_r,B_rRepr)

```
Repr ← "Yes"
if | A_r |≠| B_r | then
  Repr ← "No"
  while (A_r ≠ ∅) and (Repr="Yes") do
    let e' ∈ (e'= ≺ R, FIlls1, least1, most1, c1 ≻) ; search e 'in B_r of the same name R

    if e' ∉ B_r then        Repr← "No"

      let e' ∈ B_r(e' =≺ R, FIlls2, least2, most2, c2 ≻)

      if  (FIlls1 ≠ FIlls2) ou (least1 ≠ least2) ou (most1 ≠ most2) then
        Repr← "No"
      else//Comparison of strict sextuples and defect of normal forms of c1 and c2
        Compar(c1_θ, c2_θ, reps)
        if resp="Yes" then
          Compar(c1_δ, c2_δ, repd)
        end if

        resp ← "No"
      end if
      A_r ← A_r - e'
      B_r ← B_r - e'
      =0
```

3.6 Point de vue héritage

Dans les sections précédentes, nous avons présenté la définition de la subsomption qui signifie la classification des concepts. L'utilisateur définit ces concepts à partir des concepts primitifs et le classifieur organise automatiquement le graphe.

Le point de vue de l'héritage décrit les propriétés héritées d'un concept. Donc son objectif est de réaliser une préférence de l'exception sur le défaut.

Dans le point de vue définitionnel, i.e. le calcule de la subsomption, une exception est subsumée par le défaut, mais cette dernière n'est pas annulée par le classifieur bien qu'elle doit être rétractée par le processus d'héritage.

La procédure schéma-héritage est défini comme suit, elle consiste à trouver les propriétés héritées et à distinguer les propriétés défauts des propriétés strictes et répondre aux questions concernant les conflits et l'inconsistance.

La procédure d'héritage sert principalement à déduire l'exception à partir des dénotations de concepts (les deux parties ϵ des formes normales). L'algorithme 9 décrit la procédure de calcul d'héritage du concept C selon le scénario suivant :

1. Remplacer chaque exception de niveau pair par un défaut dans la dénotation de C.

2. Pour chaque rôle de C, appeler récursivement la procédure héritage avec restriction de valeur de rôle.
3. Supprimer P (resp. P°) dans $c_{\delta\pi}$ si P° (resp. P) est dans $c_{\theta\pi}$. La dénotation résultat est la forme d'héritage de C.

Algorithm 9 schéma-héritage [Bou11]

heritage : $CL_{\delta\epsilon} \rightarrow CL_{\delta\epsilon}$, such as

heritage (a)=res $\leftarrow \prec$ ($a_{\theta\pi}, \emptyset, \emptyset$), ($a_{\delta\pi}, \emptyset, \emptyset$) $\succ$
for all y in $a_{\theta\epsilon} \cup a_{\delta\pi}$ **do**
  res $\leftarrow$ res $\cup \prec (\emptyset, \prec$ r, héritage(p)$\succ, \emptyset), (\emptyset, \emptyset, \emptyset) \succ$
end for
for all $\prec$ r, p$\succ \in a\delta$ r **do**
  res $\leftarrow$ res $\cup \prec (\emptyset, \emptyset, \emptyset), (\emptyset, \prec$ r, héritage(p)$\succ, \emptyset) \succ$
end for//either res is $\prec$ ($res_{\theta\pi}$, $res_{\theta r}$, $res_{\theta\epsilon}$), ($res_{\delta\pi}$, $res_{\delta r}$, $res_{\delta\epsilon}$) $\succ$
for all x$^{\circ}$ in $res_{\delta\pi}$ **do**
  remove x$^{\circ}$ de $res_{\delta\pi}$
end for
Return res
=0

Les propriétés héritées sont celles trouvées dans le schéma d'héritage, les propriétés strictes (resp. propriétés défauts) sont celles qui se trouvent dans la partie stricte (resp. partie défaut).
Formellement, $re \sqsubseteq e$ *et* $re \sqsubseteq re^{inh}$, $e = e^{inh}$, mais $re^{inh} \not\sqsubseteq e^{inh}$ et $e^{inh} \not\sqsubseteq re^{inh}$. La subsomption $re \sqsubseteq re^{inh}$ signifie qu'un concept re est plus spécifique qu'un concept re^{inh} puisqu'il inclut des propriétés masquées essentielles pour avoir une classification correcte, mais ne peut pas être une propriété héritée.

Les données de la détection d'intrusion sont représentées par différentes techniques selon le besoin des recherches, on cite des prédicat de premier ordre, des modèles graphiques comme les réseaux de neurones, des modèles probabilistes comme les réseaux bayésiens, etc. Le besoin de représenter les connaissances défaut et exception et traiter le changement de contexte représentent un point essentiel dans notre recherche. Le formalisme $Jclassic_{\delta\epsilon}$ fournit cette possibilité et afin de garantir une représentation structurelle et sémantique, les données sont représentées par une ontologie où le langage $Jclassic_{\delta\epsilon}$ est utilisé comme un langage d'ontologie afin d'établir des inférences sur la pertinence des alertes.

3.7 Les Ontologies

L'introduction des ontologies dans la sécurité des systèmes informatiques revient au travail de Raskin et Nirenburg [RHTN01]. Ce travail a mis la lumière sur l'utilité des ontologies qui sont considérées comme un moyen puissant d'organiser et unifier un domaine. La contribution de ce dernier travail a ouvert la porte pour d'autres et plusieurs chercheurs dans le domaine de la sécurité informatique précisément la détection d'intrusions [UJP03] [SFLZ14].

Comme les ontologies avaient une grande popularité dans le domaine de l'informatique, on trouve différentes définitions. La définition la plus citée de terme ontologie est celle proposée dans le travail de Gruber [R93] qui signifie une représentation formelle d'un ensemble de concepts et de relations dans un domaine.

En fait, les ontologies sont utilisées pour garantir une représentation des données bien structurée, où elles fournissent au minimum la conceptualisation des vocabulaires dans un domaine. Ainsi, les ontologies permettent de clarifier la structure des connaissances et les concepts dans un domaine, ce qui améliore les systèmes de raisonnement. Ces raisons ont motivé l'utilisation des ontologies pour construire les systèmes à base de connaissances. Un système à base de connaissances se compose de trois composants importants :

- *Base de connaissances* qui est un ensemble de concepts lisibles par la machine liés à un domaine spécifique. Les connaissances sont généralement collectées auprès d'un ou de plusieurs experts du domaine et intégrées dans la base de connaissances par un ingénieur de connaissances. Les ontologies décrivent les éléments suivants :
  - Classes dans divers domaines
  - Relations entre les classes
  - Propriétés ou attributs
- *Moteur d'inférence* représente le composant qui imite le raisonnement ou l'intelligence humaine permettant à l'expert du domaine de résoudre des problèmes. IL utilise la base de connaissances pour déduire des conséquences logiques et en tirer de nouvelles conclusions. En d'autres termes, le moteur d'inférence répond aux requêtes et aux questions sur le domaine décrit dans la base de connaissances, dans lequel il observe les incohérences dans les différentes ontologies de la base de connaissances et maintient la cohérence et la mise à jour de la base de connaissances. Le moteur d'inférence fournit également les hiérarchies de classification et il calcule les inférences parmi les concepts qui ne sont pas explicitement mentionnés dans l'ontologie du domaine.
- *Interface d'utilisateur* qui est un composant d'interaction homme-machine permettant à l'utilisateur de formuler des questions et des requêtes pour le système à base de connaissances.

Il existe également de nombreux langages d'ontologies comme technique et langage de représentation des connaissances.

3.7.1 Les systèmes de détection d'intrusions à base d'ontologies

Dans la détection d'intrusions, il y a certains travaux qui ont défini formellement des ontologies pour représenter les données des intrusions. Raskin et al [RHTN01] ont introduit l'utilité des ontologies pour la sécurité informatique, ils déclarent qu'une ontologie organise et systématise tous les comportements d'intrusions à n'importe quel niveau de détail, réduisant ainsi une grande diversité d'éléments à une liste plus réduite de propriétés. Coppolino et al [CDER09] proposent une nouvelle approche, qui étend la technologie du système de détection d'intrusion (IDS) appelée technologie de détection et de diagnostic d'intrusion (ID2S). Cette approche consiste à collecter des informations à plusieurs niveaux architecturaux (Notamment : réseau, système d'exploitation, base de données et application). Le processus d'escalade des symptômes d'intrusion à la cause jugée de l'intrusion et à l'évaluation des dommages dans les composants individuels du système est déterminé par les ontologies. Une organisation hiérarchique des modèles d'événements basée sur des ontologies est utilisée pour automatiser le processus de dérivation des requêtes qui implémentent l'analyse de diagnostic. Les connaissances formalisées par l'ontologie sont utilisées par le processus de diagnostic pour identifier les intrusions/attaques susceptibles de provoquer les symptômes observés et la partie du système affectée par l'activité malveillante. Undercoffer et al [UJP03] ont construit un modèle de données qui caractérise le domaine des attaques informatique et des intrusions comme une ontologie, ils implémentent ce modèle avec un langage de représentation ontologique. Dans le travail de [ICLC09], les auteurs définissent un modèle d'ontologie pour la représentation des événements de détection et de prévention des intrusions, ainsi qu'un système hybride intelligent basé sur la mise en cluster et les réseaux neuronaux artificiels pour la classification et la reconnaissance des formes. Ils ont spécifié des attaques, des signatures, des réactions, des affirmations et des axiomes en utilisant les ontologies. Vorobiev et al [VB+10] ont proposé des ontologies pour la sécurité et les vulnérabilités, la conception de ces ontologies est basée sur l'expertise du domaine de la sécurité de l'information et des connaissances collectées provenant de diverses sources ([DNT04], [KLK05], [MSM06], [SH05]).

Le pouvoir et l'utilité des ontologies ne se réalisent pas par la simple représentation des attributs de l'attaque. Au lieu de cela, la puissance et l'utilité des ontologies sont réalisées par le fait que nous puissions exprimer les relations entre les données collectées et l'utilisation de ces relations pour en déduire que les données particulières représentent une attaque d'un type particulier.

De plus, la spécification d'une représentation ontologique dissocie le modèle de données définissant une intrusion de la logique du système de détection d'intrusion. Les ontologies

peuvent être utilisées non seulement pour fournir à un IDS la possibilité de partager une compréhension commune des informations, mais également pour lui permettre de mieux raisonner et analyser des instances de données représentant une intrusion. De plus, dans une ontologie, des caractéristiques telles que la cardinalité, l'étendue et l'exclusion peuvent être spécifiées et la notion d'héritage est supportée.

3.7.2 Langages d'ontologies

Afin de développer des ontologies et de représenter les informations qu'elles contiennent, certains langages sont proposés. Les langages les plus populaires sont RDF (Ressource Descriotion Framework), RDF schema et OWL (Ontology Web Language). Ces dernières se basent sur la représentation XML pour fournir des informations sur les objets et sur les relations entre eux.

Les langages d'ontologies permettent aux utilisateurs d'écrire explicitement les conceptualisations formelles des modèles de domaine. Les principales exigences sont :

— Une syntaxe bien définie.

— Une sémantique bien définie.

— Support de raisonnement efficace.

— Puissance d'expression suffisant.

— Commodité de l'expression.

Les langages d'ontologie devraient avoir une sémantique définie avec précision accessible par un traitement automatisé pour rendre les ontologies compréhensibles par l'homme et par la machine. OWL est recommandé par W3C (World Wide Web Consortuim) pour le web sémantique, ce langage représente un niveau de sémantique plus profond, il décrit formellement un domaine de connaissances.

SWRL (Semantic Web Rule Language) est un langage de règles combinant OWL et Rule Markup Language, il est construit afin de remédier à la faiblesse d'Owl quand il fournit un faible constructeur de rôle. SWRL permet de fournir diverses règles pour extraire les informations des ontologies. OWL fournit une représentation des informations des ontologies quand SWRL fourni des règles sous forme de queries pour extraire les informations d'ontologie.

Bien que les langages mentionnés ne permettent pas de représenter et de raisonner sur des connaissances défaut et exception, nous avons utilisé le langage $Jclassic_{\delta\epsilon}$ comme un langage d'ontologie.

$Jclassic_{\delta\epsilon}$ fournit une représentation formelle des connaissances avec une sémantique bien définie, Ainsi, $Jclassic_{\delta\epsilon}$ fournit un raisonneur qui nous permet de représenter les données défaut et exception et les règles d'inférence afin de poser des requêtes pour extraire de nouvelles connaissances représentées par l'ontologie.

3.8 Conclusion

Dans ce chapitre, nous avons brièvement introduit le raisonnement non monotone, notamment la logique de description $JClassic_{\delta\epsilon}$, qui répond aux besoins de notre contribution dans cette thèse dans une tentative pour représenter les connaissances de la détection d'intrusion.

En outre, le choix de se baser sur un formalisme est motivé par le fait que les formalismes permettent d'avoir des représentations puissantes et structurées. Ainsi qu'ils peuvent manipuler des données incertaines ou incomplètes, ce qui est le cas dans le domaine de la détection d'intrusion. Particulièrement, les systèmes d'inférence non monotone permettent de fournir un raisonnement adaptable au changement de contexte qui se produit à cause de l'arriver de nouvelles connaissances ou l'absence des autres informations.

Par ailleurs, la puissante expressivité de la logique de description non monotone $Jclassic_{\delta\epsilon}$ permet de représenter les connaissances de la détection d'intrusion et de modéliser le contexte qui caractérise la génération des alertes. Cette forme de logique nous a permis de manipuler et de représenter le changement de contexte qui peut se produire au cours du temps.

Dans la partie suivante, nous décrivons en détail notre contribution à base des informations contextuelles afin de répondre à la limite majeur des systèmes de détection d'intrusion, qui est le nombre ingérable des alertes notamment les fausses alertes.

Deuxième partie

Corrélation d'alertes et réduction du faux positif

Chapitre 4

NOC-IDS :Contextuelle ontologie avec un raisonnement non monotone

4.1 Introduction

Dans ce chapitre, nous allons présenter une approche de corrélation d'alertes à base de contexte en utilisant un raisonnement non monotone. Cette approche est appelée *NOC-IDS* (non-monotonic ontology contextual-based approach). *NOC-IDS* vise à filtrer les alertes non pertinentes qui correspondent à des objectifs d'attaques non réalisés. L'approche présentée dans cette thèse utilise les préférences de l'opérateur de sécurité afin de lui présenter les alertes qu'il veux analyser en premier lieu. Les préférences de l'opérateur de sécurité concernent beaucoup plus le choix de réseau à analyser, de cette façon, les préférences peuvent être traitées comme des informations contextuelles.

Les alertes générées sont pertinentes si les machines cibles satisfont le contexte nécessaire pour qu'elles soient affectées par les vulnérabilités exploitées. *NOC-IDS* manipule le changement de contexte dynamiquement grâce au raisonnement non monotone assuré par la logique de description $JClassic_{\delta\epsilon}$. Notre approche utilise un ensemble d'informations contextuelles sachant les types de réseau protégés, les services, les applications, les ports ouverts et les protocoles de communication. Dans notre approche, nous avons créé un ensemble de concepts et de règles d'inférence écrites en format $JClassic_{\delta\epsilon}$.

Dans le reste de ce chapitre, nous présenterons une description sur le contexte et le changement de contexte. Ensuite, nous allons présenter *NOC-IDS* en abordant l'architecture de l'approche proposée, une description de l'ontologie avec les concepts, les règles de pertinence et les étapes de processus de la corrélation. Enfin, nous expliquerons comment *NOC-IDS* assure un raisonnement dynamique et flexible.

4.2 Le contexte dans la détection d'intrusion

Une alerte fournie par un système de détection d'intrusions est souvent de très bas niveau et contient peu d'informations sur le sujet de la vulnérabilité exploitée ou de l'anomalie qu'elle décrite. Ce qui nécessite un autre type d'informations qui permet de découvrir les circonstances de l'évènement détecté. Ces informations forment le contexte de la génération des alertes. De ce fait, le contexte joue un rôle primordial dans la détection d'intrusions où chaque attaque nécessite un contexte spécifique pour se réaliser.

Le contexte est défini comme toute information qui peut caractériser la situation d'un réseau ou d'une machine dans un réseau au moment de la détection. Cette situation est décrite par un ensemble d'informations contextuelles qui incluent différents types tels que la topologie du réseau, la configuration des applications et des machines, la localisation, vulnérabilités, les utilisateurs, etc. Les informations contextuelles peuvent être collectées implicitement par des méthodes telles que le scan de vulnérabilité, empreinte de réseau (network fingerprinting), les outils passifs de surveillance des réseaux (passive network monitoring tools). Ces informations peuvent être aussi fournies par les opérateurs de sécurité via des outils tels que les systèmes de gestion de la configuration (CMS : Configuration Management Systems).
Les auteurs dans [GME09] ont classifié les informations contextuelles en quatre classes importantes.

1. Contexte relié au réseau : ce type d'information contient les informations sur la topologie du réseau, les protocoles de communication, etc. Ces informations sont utilisées par les systèmes de détection d'intrusions pour désigner les cibles et les sources des attaques.
2. Configuration de la cible : ce type d'informations concerne le système d'exploitation, les services et les applications installées dans le système cible. L'exploitation de ces informations peut être utile pour découvrir la réussite d'une attaque, où certaines attaques exploitent des vulnérabilités reliées à des applications et des services spécifiques afin d'atteindre leurs objectifs.
3. Évaluation de la vulnérabilité : ce type d'information concerne les outils de scan de vulnérabilité pour savoir si la cible est affectée par la vulnérabilité exploitée.
4. Effets des attaques : ces effets sont déterminés par l'analyse de comportement du système cible afin de découvrir la réussite d'une attaque ou son échec. Dans cette issue, une solution est proposée est celle d'utiliser un IDS comme par exemple Snort ou Bro pour analyser les réactions d'une cible à certaines attaques prédéfinies.

4.3 Le changement de contexte

La pertinence des alertes est reliée par la présence de certaines vulnérabilités dans une machine. Vu que la vulnérabilité est déterminée par certaines d'information contextuelles. Donc, le changement de contexte implique un changement sur l'état de la vulnérabilité qui conduit à une nouvelle inférence pour déduire la pertinence des alertes dans ce nouveau contexte.

Les informations contextuelles peuvent changer à cause de différentes raisons, selon le type d'information contextuelle. Pour le contexte relié au réseau, la définition des types de réseau protégés ou l'utilisation de nouveaux protocoles impliquent un changement de contexte

Le changement de la configuration des systèmes ou le traitement de certaines vulnérabilités dans le système et réseau apportent un changement de contexte. Dans notre approche, les préférences d'opérateur de sécurité sont importantes pour déterminer les sous-réseaux privilégies à protéger et les machines critiques qui nécessitent une priorité d'analyse par l'opérateur de sécurité.

4.4 *NOC-IDS* : vue générale

A la suite d'une étude que nous avons fait et qui montre l'impact de la vulnérabilité pour déterminer la pertinence des alertes [CB16], nous avons proposé une approche NOC-IDS [CB20] qui vise à associer et enrichir la sémantique des alertes et améliorer la qualité des alertes en se basant sur des informations contextuelles représentées par une ontologie, et aussi d'introduire les préférences de l'opérateur de sécurité dans le processus de la corrélation. *NOC-IDS* fournit une nouvelle représentation de données de la détection d'intrusion afin de réduire les alertes non pertinentes et aider l'opérateur de sécurité à analyser les alertes qui correspondent à ses préférences. Ce modèle profite de l'information contextuelle (application, services, nombre de port, protocole de communication et emplacement du réseau) et l'ensemble des vulnérabilités dépendantes. L'approche proposée se base sur les données contextuelles représentées sous format d'ontologie et un ensemble de règles d'inférence et une base de contexte. Cette ontologie est inspirée du modèle M4D4 [DD08] où on a modifié la définition de certains concepts pour répondre aux besoins de l'approche.

Afin de représenter les informations de l'ontologie et de poser des requêtes de pertinence des alertes, nous avons utilisé la logique de description de raisonnement non monotone $JClassic_{\delta\epsilon}$ qui permet de représenter les faits par défaut et les exceptions afin d'assurer une inférence dynamique et flexible.

Pour atteindre les objectifs de *NOC-IDS*, il est nécessaire d'identifier le contexte de différentes entités de réseau au moyen d'outils spécifiques capables d'identifier les applications

et les services existants et de détecter les risques potentiels dans les systèmes informatiques (scanners d'évaluation de la vulnérabilité, par exemple). En outre, il est nécessaire de définir les différentes informations qui caractérisent les vulnérabilités. Ces informations sont identifiées par le biais de bases de données de vulnérabilités bien connues, telles que Common Vulnerability and Exposures (CVE) [cve] et la base de vulnérabilité NVD (National Vulnerability Database) [nvd]. En plus, pour identifier la vulnérabilité exploitée par l'attaque survenue, nous pouvons interroger la base de données d'attaques Common Capack Enumeration and Classification (CAPEC) [cap].

4.4.1 Filtrer les alertes non pertinentes

Afin de réduire le nombre des alertes générées, particulièrement les faux positifs, notre approche passe par trois étapes comme il est illustré dans la figure 4.1.

Premièrement, les alertes sont standardisées au format unifié. Nous avons utilisé le format IDMEF (pour le format d'échange de messages de détection d'intrusion), qui est un modèle de données représenté dans le langage XML (Extensible Markup Language).

La base de connaissances de la logique de description (DL) est divisée en deux composants, TBOX et ABOX. TBOX contient la connaissance intentionnelle sous une forme terminologique. Elle est construite par la déclaration des propriétés des concepts. ABOX contient les connaissances d'extension, également appelées connaissances spécifiques aux individus du domaine. En deuxième étape, *NOC-IDS* consiste à pré-traiter les alertes afin de construire la base de connaissances ABOX, en fonction des informations contenues dans les alertes et des outils spécifiques permettant d'identifier les informations contextuelles.

Une fois les informations ci-dessus sont collectées et stockées dans la base ABOX à côté des différentes informations trouvées dans l'ensemble des alertes générées, le processus de la corrélation d'alertes est déclenché à la dernière étape. Le moteur de la corrélation est un ensemble de règles définies à l'aide de la logique de description non monotone $JClassic_{\delta\epsilon}$.

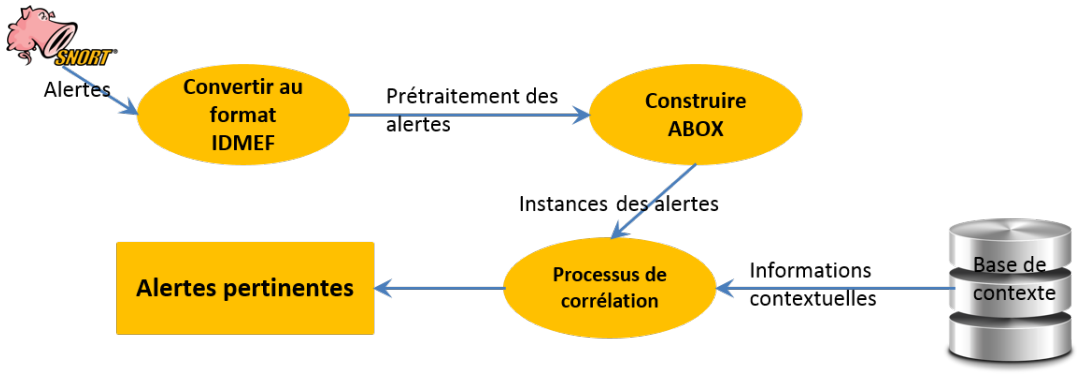

FIGURE 4.1 – Processus de filtrage des alertes non pertinentes en *NOC-IDS*

4.4.2 Concepts de *NOC-IDS* et règles d'inférence

L'ontologie utilisée dans cette approche, repose sur la relation entre quatre aspects essentiels (alerte, cible, vulnérabilité et attaque) avec un ensemble de sous-classes et de relations avec d'autres concepts comme il est montré dans la figure 4.2. La relation entre ces aspects avec l'interrogation du Contexte décide la pertinence des alertes.

Alerte. Cet aspect décrit différentes informations sur l'intrusion tels que la machine cible, la source de l'attaque, le temps de détection, la classification de l'attaque, etc. Les alertes générées sont provoquées par un ensemble d'événements survenus dans un contexte spécifique.

Cible. L'aspect cible spécifie le contexte des alertes générées. Le contexte est décrit à l'aide d'un ensemble d'informations contextuelles telles que l'adresse réseau de la machine cible, des informations sur son système d'exploitation, les applications installées et les services fournis, etc. Cet ensemble de connaissances peuvent être utiles pour déterminer les vulnérabilités affectant cette machine.

Vulnérabilité. Le fait de découvrir les vulnérabilités affectant le système, permet de réduire le risque de violation de l'information. Cet aspect décrit la vulnérabilité exploitée par l'attaque survenue. L'hôte cible est vulnérable s'il satisfait les conditions nécessaires de la vulnérabilité exploitée. Ces conditions (par exemple, le type de système d'exploitation, les applications en cours d'exécution et les services fournis) peuvent être identifiées et collectées à partir des bases de vulnérabilités telles que CVE et NVD .

Attaque. Les intrusions sont des événements survenus pour lesquels des alertes sont générées. Ces événements dépendent de deux informations principales telles que le type de l'attaque (sa classification) et la vulnérabilité exploitée. Ces informations peuvent être collectées à partir de la base de données d'attaque CAPEC.

Ces concepts sont définis pour illustrer l'avantage d'introduire des informations contextuelles pour éliminer une grande partie d'alertes inutiles. Afin d'appliquer l'ensemble des règles d'inférence et de représenter les informations contenues dans l'ontologie, l'approche *NOC-IDS* utilise la logique de description non monotone $JClassic_{\delta\epsilon}$ comme un langage d'ontologie. La logique de description $JClassic_{\delta\epsilon}$ nous permet de représenter les connaissances par défaut et exception afin de traiter le changement de contexte ce qui permet de répondre aux requêtes de pertinence lors de l'intervention de de nouvelles connaissance sur le contexte sans avoir refaire l'inférence du le début. La base de connaissances *TBOX* comprend quatre concepts principaux tels que *cible*, *alerte*, *attaque* et *vulnérabilité*. Nous représentons maintenant les attributs qui définissent chaque concept.

Cible. La machine cible est décrite par son environnement, ce dernier est considéré comme le contexte dans lequel l'événement est survenu. Le contexte de la machine cible est déterminé par toute information qui peut décrire la situation d'une machine ou d'un réseau (les services et les applications, type de réseau, les utilisateurs, le système d'exploitation et sa version, etc.). Dans ce cas, le concept cible représente le contexte dans lequel

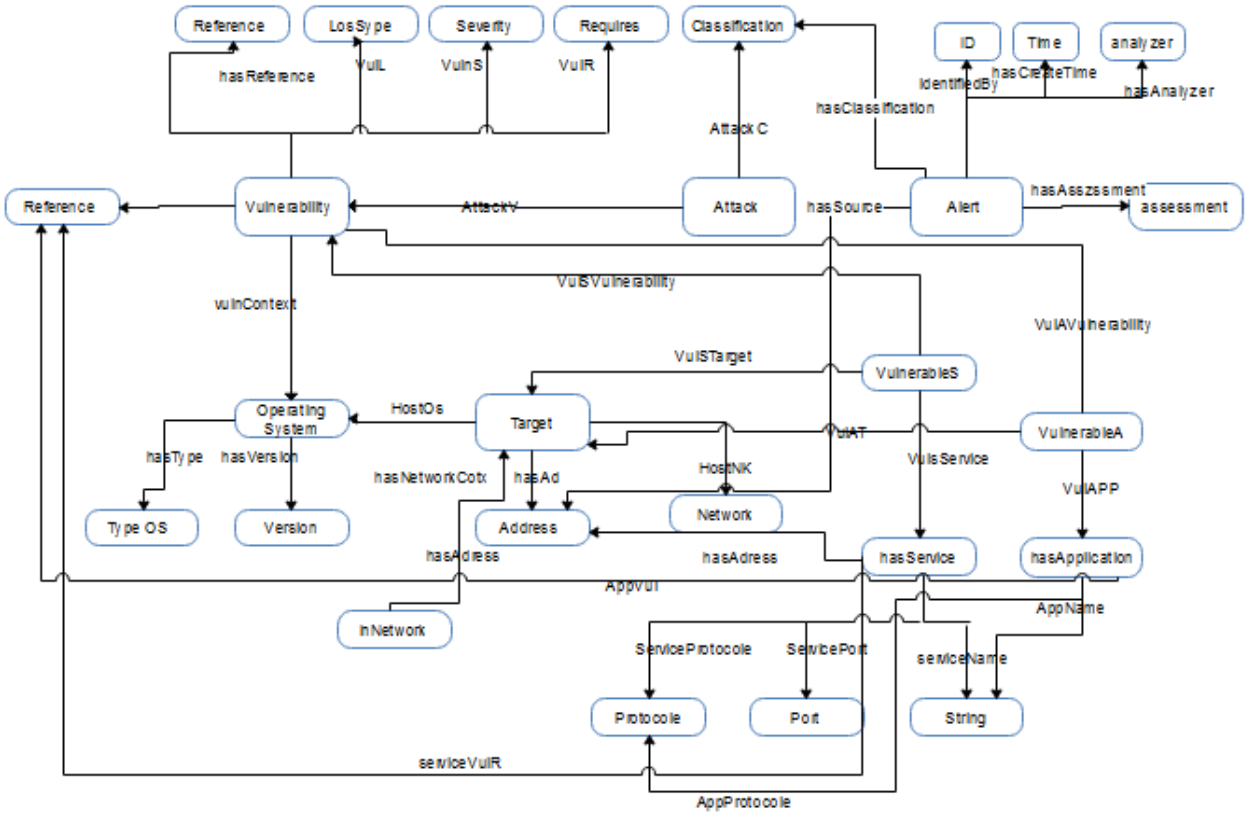

FIGURE 4.2 – *NOC-IDS* ontologie pour la corrélation d'alertes

les alertes se produisent. Le modèle M4D4 utilise un ensemble d'attributs pour identifier un hôte dans un réseau tel que l'adresse IP de l'hôte et l'adresse IP du réseau. Dans notre modèle, nous avons utilisé l'adresse IP de la cible et le type de réseau qu'elle appartient, ainsi nous avons ajouté le système d'exploitation et sa version. Les information sur les applications et les services fournis sur cet hôte sont déduites par des règles d'inférence qu'on va représenter ultérieurement. Les informations sur le système d'exploitation sont représentées dans l'ontologie par le concept *OperatingSystem*.

$OperatingSystem \sqsubseteq \forall\ hasType : TypeOS \sqcap hasType\ AT\text{-}LEAST\ 1 \sqcap \forall\ hasVersion : Version \sqcap hasVersion\ AT\text{-}LEAST\ 1 \sqcap \forall\ hasOData : OtherData$
Le concept *OperatingSystem* décrit le système d'exploitation utilisé. Il est identifié par le type de système d'exploitation (TypeOS), sa version et le champ *OtherData* (ce champ est réservé pour décrire une autre information à la demande). Les relation *hasType*, *hasVersion* et *hasOData* sont des relations binaires qui relient respectivement :
Le concept *OperatingSystem* avec *TypeOs*
Le concept *OperatingSystem* avec *Version*
Le concept *OperatingSystem* avec *OtherData*

La base de connaissances TBOX contient les axiomes suivants :
$Address \sqsubseteq \top$
$Network \sqsubseteq \top$
$TypeOS \sqsubseteq \top$
$OtherData \sqsubseteq \top$
$Version \sqsubseteq \top$

$Target \sqsubseteq \forall\ HostAd : Address \sqcap HostAd\ AT\text{-}LEAST\ 1 \sqcap \forall\ HostNk : Network \sqcap HostNK\ AT\text{-}LEAST \sqcap \forall\ HostOs : OperatingSystem \sqcap HostOS\ AT\text{-}LEAST\ 1$
Cette axiome représente chaque machine cible dans un réseau, où les relations binaires *HostAd, HostNk, and 0HostOs* relient respectivement :
Le concept target avec le concept Address.
Le concept target avec le concept Network.
Le concept target avec le concept OperatingSystem.
Alert. Ce concept représente des informations générales sur l'intrusion. Le standard IDMEF (Intrusion Detection Message Exchange Format) représente un format unifié écrit en langage XML pour représenter les alertes générées par les IDSs. Afin de représenter les alertes, nous utilisons les attributs fournis par le standard IDMEF tels que l'identificateur d'une alerte, l'heure de création de l'alerte, le temps de la détection, l'heure actuelle de l'analyseur, l'analyseur qui génère l'alerte, l'adresse IP de l'intrus, l'adresse IP cible, la classification de l'intrusion ou son nom, la gravité de l'attaque indiquée par le champ

assessment et le champ *other data* qui est réservé pour décrire tout autre type d'information. L'axiome suivant illustre comment nous avons présenté le concept *Alert* en $JClassic_{\delta\epsilon}$:

$$
\begin{aligned}
&Alert \sqsubseteq \forall\ identifiedBY.ID \sqcap \\
&= 1\ identifiedBy \sqcap \\
&hasCreateTime.Time \sqcap \\
&= 1\ hasCreateTime \sqcap \\
&\forall\ hasDetectTime.Time \sqcap \\
&hasDetectTime\ AT\text{-}MOST\ 1 \sqcap \\
&\forall\ hasAnalyserTime.Time \sqcap \\
&hasAnalyserTime\ AT\text{-}MOST\ 1 \sqcap \\
&\forall\ hasAnalyser.Analyser \sqcap \\
&= 1\ hasAnalyser \sqcap \\
&\forall\ hasSource.Source \sqcap \\
&\forall\ hasTarget.Target \sqcap \\
&\forall\ hasClassification.Classification \sqcap \\
&= 1\ hasClassification \sqcap \\
&\forall\ hasAssessement.Assessment \sqcap \\
&hasAssessment\ AT\text{-}MOST\ 1 \sqcap \\
&\leq hasAdditionalData.AdditionalData
\end{aligned}
$$

La base de connaissance TBOX contient les axiomes suivantes :

$ID \sqsubseteq \top$
$Time \sqsubseteq \top$
$Analyser \sqsubseteq \top$
$Classification \sqsubseteq \top$
$Assessment \sqsubseteq \top$
$AdditionalData \sqsubseteq \top$
$Source \sqsubseteq SourceNode.Node \sqcap SourceUser.User \sqcap SourceProcess.Process \sqcap SourceService.Service$
La relation binaire *hasTarget* relie le concept *Alert* avec le concept *Target*.

Vulnérabilité. Ce concept décrit formellement les vulnérabilités qui affectent la machine. Le modèle M4D4 fournit une définition de la vulnérabilité par un ensemble d'attributs tels que le degré de gravité qui prend une valeur de faible, moyen ou élevé, le niveau d'accès nécessaire si elle est exploitée, le type d'information violée, et sa date de publication.

NOC-IDS définit le concept vulnérabilité en incluant les attributs suivants :
la référence de la vulnérabilité qui est utilisé par plusieurs outils de sécurité pour décrire leur observation et l'attribut contexte (ici, il est représenté par le concept *OperatingSystem*). L'attribut contexte décrit les conditions requises pour que la machine soit vulnérable.
La base de connaissances *TBOX* contient les axiomes suivants :
$Reference \sqsubseteq \top$
$Severity \sqsubseteq \top$
$Requires \sqsubseteq \top$
$LossType \sqsubseteq \top$
$PublishedDate \sqsubseteq \top$
Nous proposons la définition du concept *Vulnerability* en $JClassic_{\delta\epsilon}$ comme suit :
$Vulnerability \sqsubseteq \forall\ hasReference : Reference \sqcap hasReference\ AT\text{-}LEAST\ 1 \sqcap \forall\ VulnS : Severity \sqcap VulnS\ FILLS\ \{low, medium, high\} \sqcap \forall\ VulnR : Requires \sqcap VulnR\ FILLS\ \{remote, local, user\} \sqcap \forall\ VulnL : LossType \sqcap VulnL\ FILLS\ \{confidentiality, integrity, availability, privilege\_escalation\} \sqcap \forall\ VulnP : PublishedDate \sqcap VulnP\ AT\text{-}LEAST\ 1 \sqcap vulnContext : OperatingSystem.$
Les relations binaires *hasReference, VulnS, VulnR, VulnL, VulnP, vulnContext* relient respectivement le concept *Vulnerability* par *Reference*, *Severitydegree, Requires, LossType, publisheddate*, et *OperetingSystem.*

Attaque. La *TBOX* contient également le concept attaque pour décrire les événements qui sont considérés comme des intrusions. Dans [SFLZ14], les auteurs ont défini l'intrusion par son objectif, alors que dans *NOC-IDS*, on représente chaque attaque par sa classification et la vulnérabilité exploitée. La base de connaissances *TBOX* contient les axiomes suivants :
$Classification \sqsubseteq \top$
$Attack \sqsubseteq \forall\ AttackC : Classification \sqcap AttackC\ AT\text{-}LEAST\ 1 \sqcap \forall\ AttackV : Vulnerability \sqcap AttackV\ AT\text{-}LEAST\ 1.$
L'axiome *Attack* illustre la représentation du concept *Attack* en $JClassic_{\delta\epsilon}$, où les relations binaires *AttackC* et *AttackV* relient respectivement le concept *Attack* avec les concepts *classification* et *vulnerability.*

Le tableau 6.2 résume la base de connaissances terminologiques TBOX. Ce vocabulaire contient la définition des concepts et sous forme des axiomes terminologiques.

$\boldsymbol{OperatingSystem} \sqsubseteq \forall\ hasType : TypeOS \sqcap hasType\ AT\text{-}LEAST\ 1 \sqcap$ $\forall\ hasVersion : Version \sqcap hasVersion\ AT\text{-}LEAST\ 1 \sqcap \forall\ hasOData : OtherData$
$\boldsymbol{Address} \sqsubseteq \top$
$\boldsymbol{Network} \sqsubseteq \top$
$\boldsymbol{TypeOS} \sqsubseteq \top$
$\boldsymbol{OtherData} \sqsubseteq \top$

Version $\sqsubseteq \top$
Reference $\sqsubseteq \top$
Severity $\sqsubseteq \top$
Requires $\sqsubseteq \top$
LossType $\sqsubseteq \top$
PublishedDate $\sqsubseteq \top$
Classification $\sqsubseteq \top$
Port $\sqsubseteq \top$
Protocol $\sqsubseteq \top$
ID $\sqsubseteq \top$
Time $\sqsubseteq \top$
Analyser $\sqsubseteq \top$
Assessment $\sqsubseteq \top$
AdditionalData $\sqsubseteq \top$
Target $\sqsubseteq \forall$ *HostAd : Address* $\sqcap$ *HostAd AT-LEAST 1* $\sqcap \forall$ *HostNk :Network*
$\sqcap$ *HostNK AT-LEAST* $\sqcap \forall$ *HostOs : OperatingSystem* $\sqcap$ *HostOS AT-LEAST 1*
Alert $\sqsubseteq \forall$ *messageId.String* $\sqcap =$ *1 messageId* $\sqcap$ *hasCreateTime.Time* $\sqcap = 1$ *hasCreateTime*
$\sqcap \forall$ *hasDetectTime.Time* $\sqcap$ *hasDetectTime AT-MOST 1*$\sqcap \forall$ *hasAnalyserTime.Time* $\sqcap$
hasAnalyserTime AT-MOST 1$\sqcap \forall$ *hasAnalyser.Analyser* $\sqcap =$ *1 hasAnalyser*
$\sqcap \forall$ *hasSource.Source* $\sqcap \forall$ *hasTarget.Target* $\sqcap \forall$ *hasClassification.Classification*
$\sqcap =$ *1 hasClassification* $\sqcap \forall$ *hasAssessement.Assessment* $\sqcap$ *hasAssessment AT-MOST 1*
$\sqcap \leq$ *hasAdditionalData.AdditionalData.*
Source $\sqsubseteq$ *SourceNode.Node* $\sqcap$ *SourceUser.User* $\sqcap$ *SourceProcess.Process* $\sqcap$*SourceService.Service*
Vulnerability $\sqsubseteq \forall$ *hasReference : Reference* $\sqcap$ *hasReference AT-LEAST 1* $\sqcap \forall$ *VulnS :Severity*
$\sqcap$ *VulnS FILLS* {*low,medium,high*} $\sqcap \forall$ *VulnR :Requires* $\sqcap$ *VulnR FILLS* {*remote, local, user*}
$\sqcap \forall$ *VulnL :LossType* $\sqcap$ *VulnL FILLS* {*confidentiality, integrity, availability, privilege_ escalation*}
$\sqcap \forall$ *VulnP :PublishedDate* $\sqcap$ *VulnP AT-LEAST 1* $\sqcap$ *vulnContext : OperatingSystem.*
Attack $\sqsubseteq \forall$ *AttackC :Classification* $\sqcap$ *AttackC AT-LEAST 1* $\sqcap \forall$ *AttackV :Vulnerability* $\sqcap$
AttackV AT-LEAST 1.
hasService $\sqsubseteq \forall$ *serviceName :String* $\sqcap$ *serviceName AT-LEAST 1* $\sqcap \forall$ *hasAddress :Address* $\sqcap$
hasAddress AT-LEAST 1 $\sqcap \forall$ *servicePort :Port* $\sqcap$ *servicePort AT-LEAST 1* $\sqcap$ *ServiceProtocole :*
Protocole $\sqcap$ ServiceProtocole FILLS of { TCP, UDP,IP, ICMP } $\sqcap \forall$ serviceVulR :Reference
$\sqcap$ *serviceVulR AT-LEAST 1* .
δ ***VulnerableS*** $\sqsubseteq \forall$ *VulSTarget :Target* $\sqcap$ *VulSTarget AT-LEAST 1* $\sqcap \forall$ *VulSService : hasService*
$\sqcap$ *VulSService AT-LEAST 1* $\sqcap \forall$ *VulSVulnarebility : Vulnerability*
$\sqcap$ *VulSVulnarebility AT-LEAST 1.*
δ ***relevantAlert2*** $\sqsubseteq \forall$ *IdAlert :Alert* $\sqcap$ *IdAlert AT-LEAST 1* $\sqcap \forall$ *RlvAlrtA :Attack* $\sqcap$
RlvAlrtA AT-LEAST 1$\sqcap$ *RlvAlrtT : Target* $\sqcap$ *RlvAlrtT AT-LEAST 1* $\sqcap \delta$ *VulnerableS*

hasApplication $\sqsubseteq \forall$ *AppName :String* $\sqcap$ *AppName AT-LEAST 1* $\forall$ *AppProtocol :Protocol* $\sqcap$ *AppProtocol AT-LEAST 1* $\sqcap\forall$*AppVul :Reference* $\sqcap$ *AppVul AT-LEAST 1* .
δ ***VulnerableA*** $\sqsubseteq$ *VulAPP :hasApplication* $\sqcap$ *VulAPP AT-LEAST 1* $\sqcap\forall$ *VulAT :Target* $\sqcap$ *VulAT AT-LEAST 1* $\sqcap\forall$ *VulnAVulnerability : Vulnerability* $\sqcap$ *VulnAVulnerability AT-LEAST 1* .
δ ***ReleventAlert1*** $\sqsubseteq \forall$ *IdAlert :Alert* $\sqcap$ *IdAlert AT-MOST 1* $\sqcap\forall$ *RlvAlrtA : Attack* $\sqcap$ *RlvAlrtA AT-LEAST 1* $\sqcap$ *RlvAlrtT :Target* $\sqcap$ *RlvAlrtT AT-LEAST 1* $\sqcap\delta$ *InNetwork* .
$(\delta$ ***ReleventAlert1*** $)^{\epsilon} \sqsubseteq \forall$ *IdAlert :Alert* $\sqcap$ *IdAlert AT-MOST 1* $\sqcap\forall$ *RlvAlrtA : Attack* $\sqcap$ *RlvAlrtA AT-LEAST 1* $\sqcap$ *RlvAlrtT :Target* $\sqcap$ *RlvAlrtT AT-LEAST 1* $\sqcap$ $(\delta$ *InNetwork*$)^{\epsilon}$.
δ ***relevantAlert2*** $\sqsubseteq \forall$ *IdAlert :Alert* $\sqcap$ *IdAlert AT-LEAST 1* $\sqcap\forall$ *RlvAlrtA :Attack* $\sqcap$ *RlvAlrtA AT-LEAST 1*$\sqcap$ *RlvAlrtT : Target* $\sqcap$ *RlvAlrtT AT-LEAST 1* $\sqcap\delta$ *VulnerableS*
$(\delta$ ***relevantAlert2***$)^{\epsilon} \sqsubseteq \forall$ *IdAlert :Alert* $\sqcap$ *IdAlert AT-LEAST 1* $\sqcap\forall$ *RlvAlrtA :Attack* $\sqcap$ *RlvAlrtA AT-LEAST 1*$\sqcap$ $(\delta$ *hasServiceVul(String)*$)^{\epsilon}$.
δ ***relevantAlert3*** $\sqsubseteq \forall$ *IdAlert :ID* $\sqcap$ *IdAlert AT-LEAST 1* $\sqcap\forall$ *RlvAlrtA :Attack* $\sqcap$ *RlvAlrtA AT-LEAST 1* $\sqcap$ *RlvAlrtT : Target* $\sqcap$ *RlvAlrtT AT-LEAST 1* $\sqcap\delta$ *VulnerabilityA*
$(\delta$ ***relevantAlert3***$)^{\epsilon} \sqsubseteq \forall$ *IdAlert :ID* $\sqcap$ *IdAlert AT-LEAST 1* $\sqcap\forall$ *RlvAlrtA :Attack* $\sqcap$ *RlvAlrtA AT-LEAST 1* $\sqcap$ $(\delta$ *VulnerabilityA*$)^{\epsilon}$.

TABLE 4.1 – Résumé de la base des connaissances terminologique TBOX

Dans la sous-section suivante, nous allons présenter comment *NOC-IDS* traite le contexte et le changement de contexte en appliquons certaines règles d'inférence qui permettent de mettre en évidence la relation entre les aspects de l'ontologie.

4.4.3 Règles d'inférence de *NOC-IDS*

Rappelons que l'un des objectifs de l'approche *NOC-IDS* est d'assurer une inférence plus dynamique et flexible de telle manière à fournir à l'opérateur de sécurité la possibilité de protéger le réseau conformément à ses intérêts et ses préférences. Par conséquent, l'approche proposée utilise un ensemble d'informations contextuelles pour identifier les conditions nécessaires pour être vulnérable.

Dans le système non monotone, si un fait positif n'est pas déduit de la base de connaissances, alors il est faux. C'est pourquoi, il suffit de représenter que les faits positifs. De ce fait, nous définissons dans notre modèle les règles qui amènent à déduire les alertes pertinentes. Dans le cas où la cible ne comporte pas la vulnérabilité exploitée, l'alerte correspondante sera rédigée vers un fichier journal, ce qui réduit le nombre d'alertes à analyser.

Dans cette section, nous illustrons comment *NOC-IDS* traite les changements de contexte dans la corrélation d'alertes afin d'inférer les alertes pertinentes. Ces dernières sont représentées par la règle *IS_ RelevantAlert* qui se base sur un ensemble de concepts

bien définis notés δ *ReleventAlert*$_i$ (i est un indice pour représenter les informations contextuelles) décrivant la pertinence par défaut d'une alerte selon une information contextuelle spécifique.

En général, la règle *IS_ RelevantAlert* est définie comme une séquence de δ *RelevantAlerti* comme suit :
Is_ relevantAlert $\sqsubseteq$ δ *RelevantAlerti*

Dans le cas de changement du contexte, les propriétés par défaut qui définissent le contexte par défaut sont des exceptions dans le concept δ *RelevantAlert*$_i$.

Selon l'algorithme de subsomption de *JClassic*$_{\delta\epsilon}$, l'exception d'une propriété défaut est décrite comme suit :

$A^{\epsilon} \sqsubseteq (\delta\ A)^{\epsilon}$. Ce qui implique la déduction de la subsomption suivante :
(RelevantAlerti)$^{\epsilon}$ $\sqsubseteq$ (δ *RelevantAlerti)*$^{\epsilon}$
Et vue que dans *JClassic*$_{\delta\epsilon}$, on ne peut pas déduire à partir d'une exception :
Is_ relevantAlert $\not\sqsubseteq$ *(RelevantAlerti)*$^{\epsilon}$.
Par conséquent, le système ne peut pas déduire l'alerte correspondante comme pertinente.

Type de réseau protégé. Plusieurs attaques ciblent un sous-réseau bien défini. Connaître le sous-réseau cible est considéré comme une information contextuelle, ce qui a un rôle important à jouer pour déterminer si l'alerte est pertinente ou faux positif. L'idée est fondamentale. Premièrement, l'opérateur de sécurité identifie le sous-réseau qui l'intéresserait. Ensuite, chaque alerte générée à la suite d'un événement donné qui cible un hôte quelconque de ce sous-réseau est pertinente et doit être analysée par l'opérateur de sécurité pour vérifier le succès de l'attaque.

Nous pouvons utiliser l'adresse IP cible d'information trouvée dans l'alerte pour déterminer le réseau cible. Si la machine cible est par défaut sur le réseau que l'opérateur de sécurité souhaite protéger, l'alerte générée est par défaut pertinente. De manière formelle, nous définissons le concept δ *InNetwork* comme suit :
δ *InNetwork* $\sqsubseteq$ $\forall$ *hasAddress :Address* $\sqcap$ *hasAddress AT-LEAST 1* $\sqcap\forall$ *hasNetwork-Cotx :Target* $\sqcap$ *hasNetworkCotx AT-LEAST 1.*

Cela suffit pour dire que l'alerte générée est pertinente par défaut, elle est représentée par le concept δ *ReleventAlert1* comme suit :

δ *ReleventAlert1* $\sqsubseteq$ $\forall$ *IdAlert :Alert* $\sqcap$ *IdAlert AT-MOST 1* $\sqcap\forall$ *RlvAlrtA : Attack* $\sqcap$ *RlvAlrtA AT-LEAST 1* $\sqcap$ *RlvAlrtT :Target* $\sqcap$ *RlvAlrtT AT-LEAST 1* $\sqcap\delta$ *InNetwork .*

La règle δ *ReleventAlert1* indique qu'une alerte est pertinente par défaut si elle est générée à la suite d'une attaque, dans laquelle cette attaque cible un hôte appartenant par

défaut au réseau que l'opérateur de sécurité souhaite protéger.

En appliquant la règle de pertinence *Is_ relevantAlert*, nous pouvons en déduire que l'alerte générée est pertinente.

Is_ relevantAlert $\sqsubseteq$ δ *ReleventAlert1*

Dans le cas où le contexte est modifié, nous introduisons la notion d'exception. Plus précisément, et pour le même type d'attaque ; l'hôte cible appartient à un réseau qui ne se trouve pas dans la zone d'intérêt de l'opérateur de sécurité. Nous présentons ce cas avec la règle suivante :

(δ *ReleventAlert1)*$^{\in}$ $\sqsubseteq$ $\forall$ *IdAlert :Alert* $\sqcap$ *IdAlert AT-MOST 1* $\sqcap\forall$ *RlvAlrtA : Attack* $\sqcap$ *RlvAlrtA AT-LEAST 1* $\sqcap$ *RlvAlrtT :Target* $\sqcap$ *RlvAlrtT AT-LEAST 1* $\sqcap$ *(*δ *InNetwork)*$^{\in}$.

Dans ce cas, le contexte est relié beaucoup plus à la zone d'intérêt de l'opérateur de sécurité. L'exception de δ *InNetwork*, permte de ne pas sélectionner l'alerte :

Is_ relevantAlert $\not\sqsubseteq$ *ReleventAlert1* $^{\in}$

Service cible. Lorsqu'un intrus tente de cibler un hôte fournissant un service spécifique et que cet hôte satisfait aux conditions nécessaires pour être affecté par cette vulnérabilité de service, les alertes correspondantes sont considérées comme pertinentes par défaut. Tout d'abord, nous définissons le concept suivant pour déterminer si l'hôte cible fournit un service donné spécifiant son port, le protocole utilisé et la référence de la vulnérabilité exploitée :

hasService $\sqsubseteq$ $\forall$ *serviceName :String* $\sqcap$ *serviceName AT-LEAST 1* $\sqcap\forall$ *hasAddress :Address* $\sqcap$ *hasAddress AT-LEAST 1* $\sqcap\forall$ *servicePort :Port* $\sqcap$ *servicePort AT-LEAST 1* $\sqcap$*ServiceProtocole :Protocole*$\sqcap$ *ServiceProtocole FILLS of { TCP, UDP,IP, ICMP }* $\sqcap\forall$ *serviceVulR :Reference* $\sqcap$ *serviceVulR AT-LEAST 1* .

Le concept *hasService* indique si un hôte donné (identifié par son adresse IP) fournit un service donné (représenté par son port). Ce concept désigne également la vulnérabilité associée à ce service (représentée par sa référence).

Ensuite, nous définissons le concept δ *Vulnerable* pour spécifier si la machine cible est par défaut sous le contexte considérée vulnérable. Avec un autre mot, ce concept spécifie si la machine cible fournit le service cible tout en satisfaisant le contexte pour être vulnérable. La représentation de l'hôte vulnérable par défaut est comme suit :

δ *VulnerableS* $\sqsubseteq$ $\forall$ *VulSTarget :Target* $\sqcap$ *VulSTarget AT-LEAST 1* $\sqcap\forall$ *VulSService : hasService* $\sqcap$ *VulSService AT-LEAST 1* $\sqcap\forall$ *VulSVulnarebility : Vulnerability* $\sqcap$ *VulSVulnarebility AT-LEAST 1.*

Le concept δ *VulnerableS* signifie que l'hôte cible est vulnérable par défaut. Cet hôte fournit un service donné et satisfait le contexte nécessaire pour avoir cette vulnérabilité de service. Les informations sur le système d'exploitation utilisé sont exprimées dans le concept Target.

Une alerte est pertinente par défaut si elle est générée à la suite d'une attaque visant un hôte vulnérable par défaut en exploitant une vulnérabilité spécifique. Cet hôte cible est inféré par δ *VulnerableS*. Ainsi, nous définissons le concept suivant pour définir l'alerte pertinente en fonction de la vulnérabilité du service d'informations contextuelles :
δ *relevantAlert2* $\sqsubseteq \forall$ *IdAlert :Alert* $\sqcap$ *IdAlert AT-LEAST 1* $\sqcap\forall$ *RlvAlrtA :Attack* $\sqcap$ *RlvAlrtA AT-LEAST 1*$\sqcap$ *RlvAlrtT : Target* $\sqcap$ *RlvAlrtT AT-LEAST 1* $\sqcap\delta$ *VulnerableS*

La règle δ *relevantAlert2* signifie que l'alerte générée est pertinente par défaut. Cette alerte est générée à la suite d'une attaque donnée dont l'objectif est de compromettre un service fourni par l'hôte cible, et cet hôte est vulnérable par défaut (il remplit la condition nécessaire pour être affecté par la vulnérabilité exploitée). À partir de là, nous pouvons appliquer la règle de pertinence comme suit :

Is_ relevantAlert $\sqsubseteq \delta$ *ReleventAlert2*
L'état du service de vulnérabilité peut être modifié avec le temps. Cette modification est due aux vulnérabilités qui ont été supprimées à l'aide de correctifs logiciels ou à l'apparition de nouvelles vulnérabilités dans le système [Goe16]. Dans le modèle proposé, l'alerte générée devient sans importance si elle est générée à la suite d'un événement donné qui cible un service particulier, dans lequel son état vulnérable a été modifié. On peut le représenter par la règle suivante :
(δ *Is_ relevantAlert)*$^{\epsilon}$ $\sqsubseteq \forall$ *IdAlert :Alert* $\sqcap$ *IdAlert AT-LEAST 1* $\sqcap\forall$ *RlvAlrtA :Attack* $\sqcap$ *RlvAlrtA AT-LEAST 1*$\sqcap$ *(*δ *hasServiceVul(String))*$^{\epsilon}$.
Ainsi, l'alerte correspondante ne sera pas déduite en tant qu'alerte pertinente.
Is_ relevantAlert $\not\sqsubseteq \delta$ *ReleventAlert2.*

Application cible. L'attaquant peut cibler une application particulière pour obtenir son intrusion en exploitant la vulnérabilité liée à cette application. Afin de définir les alertes pertinentes en fonction de la vulnérabilité de l'application, nous définissons le concept suivant
hasApplication $\sqsubseteq \forall$ *AppName :String* $\sqcap$ *AppName AT-LEAST 1* $\forall$ *AppProtocol :Protocol* $\sqcap$ *AppProtocol AT-LEAST 1* $\sqcap\forall$*AppVul :Reference* $\sqcap$ *AppVul AT-LEAST 1* .

Le concept *hasApplication* vérifie si un hôte cible donné exécute l'application représentée par son nom (type de chaîne), son protocole et sa référence de vulnérabilité. Pour identifier le contexte sous-jacent de vulnérabilité d'application, nous définissons le concept

suivant :

δ *VulnerableA* $\sqsubseteq$ *VulAPP :hasApplication* $\sqcap$ *VulAPP AT-LEAST 1* $\sqcap\forall$ *VulAT :Target* $\sqcap$ *VulAT AT-LEAST 1* $\sqcap\forall$ *VulnAVulnerability : Vulnerability* $\sqcap$ *VulnAVulnerability AT-LEAST 1* .

La règle δ*VulnerableA* indique que l'hôte est vulnérable par défaut s'il exécute l'application ciblée tout en satisfaisant le contexte pour contenir la vulnérabilité associée.

À partir de là, nous supposons qu'une alerte est pertinente si l'attaque cible un hôte, qui est vulnérable par défaut à la vulnérabilité de l'application exploitée telle qu'elle est définie dans le concept : δ *relevantAlert3*.
δ *relevantAlert3* $\sqsubseteq \forall$ *IdAlert :ID* $\sqcap$ *IdAlert AT-LEAST 1* $\sqcap$
$\forall$ *RlvAlrtA :Attack* $\sqcap$ *RlvAlrtA AT-LEAST 1* $\sqcap$ *RlvAlrtT : Target* $\sqcap$ *RlvAlrtT AT-LEAST 1* $\sqcap\delta$ *VulnerabilityA*
La règle δ *relevantAlert3* signifie qu'une alerte est pertinente si elle est générée à la suite d'une attaque qui tente de compromettre une application donnée. Cette attaque exploite la vulnérabilité associée pour atteindre son objectif et que l'hôte cible exécute cette application dans laquelle il est vulnérable par défaut.

À partir de cela, nous pouvons appliquer la règle de pertinence *Is_ relevantAlert* pour déduire l'ensemble des alertes pertinentes en fonction de l'application cible.

Is_ relevantAlert $\sqsubseteq \delta$ *relevantAlert3*
Une fois le contexte est modifié, les faits par défaut deviennent des exceptions au sens de la règle suivante :
(δ *relevantAlert3)*$^{\epsilon}$ $\sqsubseteq \forall$ *IdAlert :ID* $\sqcap$ *IdAlert AT-LEAST 1* $\sqcap\forall$ *RlvAlrtA :Attack* $\sqcap$ *RlvAlrtA AT-LEAST 1* $\sqcap$ *(*δ *VulnerabilityA)*$^{\epsilon}$.

À l'aide de cet ensemble de règles, l'algorithme de subsomption permet d'inférer l'ensemble des alertes pertinentes en fonction du contexte initial.

Nous devrions dire que le modèle proposé fonctionne comme un outil complémentaire du système de détection d'intrusion (dans notre étude, nous utilisons Snort). Ainsi, les règles proposées ne sont pas intégrées à l'IDS, mais elles constituent un complément qui peut aider l'opérateur de sécurité à filtrer les alertes faux positifs et à analyser les pertinentes.

4.5 Conclusion

Dans ce chapitre, nous avons présenté l'approche *NOC-IDS* qui est une approche de corrélation d'alertes vise à fournir un modèle formel afin de réduire le nombre d'alertes à analyser et plus précisément le nombre d'alertes non pertinentes. *NOC-IDS* repose sur

un raisonnement non monotone qui permet de modéliser le changement de contexte afin de surmonter les limites des techniques de corrélation d'alertes existant dans la littérature.

Nous avons abordé le filtrages des alertes ainsi la découverte de la stratégie des attaques . Dans la phase de filtrage des alertes, nous focalisons sur l'élimination des alertes non pertinentes. Nous avons utilisé le contexte et le changement de contexte pour assurer une technique de corrélation plus robuste, nous avons proposé de représenter les connaissances de la détection d'intrusions par une ontologie où $JClassic_{\delta\in}$ fournit un langage d'ontologie. $JClassic_{\delta\in}$ permet d'extraire les informations contenues dans l'ontologie et de représenter l'ensemble des règles d'inférence afin de déduire les alertes pertinentes.

La découverte de la stratégie des attaques repose sur les alertes inférées comme pertinentes pour découvrir les relations possibles entre eux en utilisant certains attributs existant dans les alertes.

Dans le chapitre suivant, nous allons présenter l'implémentation de l'approche *NOC-IDS* en abordant quelques problèmes d'évaluation.

Chapitre 5

Implémentation et évaluation

5.1 Introduction

Ce chapitre se compose de deux parties. Dans la première, nous décrivons l'implémentation de l'ontologie proposée dans le chapitre 4 et comment le raisonnement se fait à partir d'un ensemble de règles d'inférence. La deuxième partie décrit les problèmes d'évaluation des approches de la corrélation d'alertes, où le majeur problème est le non traitement des changements de contexte et le manque de description du contexte de l'environnement. Dans une tentative pour répondre à ces besoins, nous avons généré une instance d'attaque du type Island hopping. Dans cette partie, nous décrivons les outils et les techniques employées pour générer cette attaque.

5.2 Implémentation de l'ontologie NOC-IDS

L'ontologie proposée dans le chapitre 4 consiste d'un ensemble de concepts et de relations entre eux. Cette ontologie est implémentée en utilisant l'outil $T\_JClassi\delta\epsilon$ [KG18], qui représente une version développée et améliorée afin de manipuler des connaissances temporelles non monotones.

$T\_JClassic\delta\epsilon$ permet de créer des bases de connaissances, d'ajouter, de modifier et de supprimer des concepts, des rôles, et des individus d'un domaine donné. Il garantit des services d'inférences usuelles qui se trouvent généralement dans les moteurs d'inférence pour raisonner et déduire de nouvelles connaissances, la figure 5.1 représente l'interface graphique de l'outil $T\_JClassic\delta\epsilon$.

L'ontologie se compose d'un ensemble de concepts primitifs, d'autres composés (voir figure 5.2) et d'un ensemble de rôles (voir la figure 5.3). Les instances sont remplies à partir d'un ensemble de programmes développés en java permettent de remplir la ABOX à partir des fichiers XML concernant les alertes et les informations contextuelles où les préférences de l'opérateur de sécurité sont remplis manuellement.

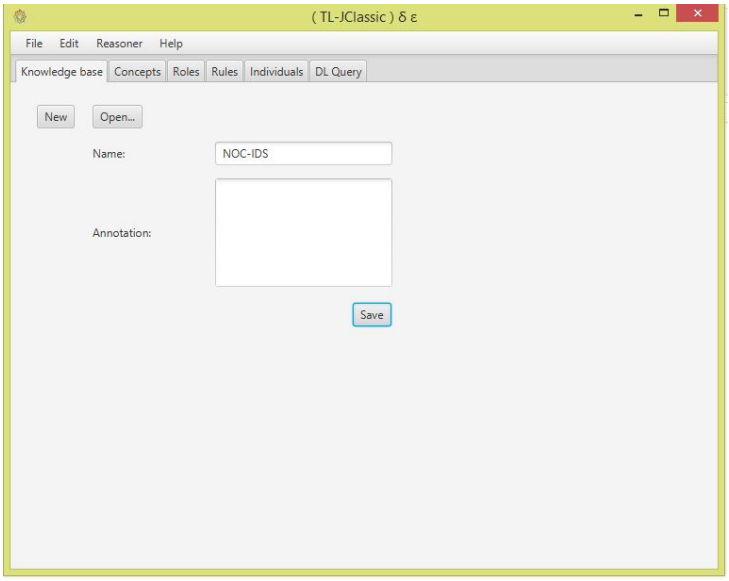

FIGURE 5.1 – L'interface graphique de l'outil $T\_JClassic\delta\epsilon$

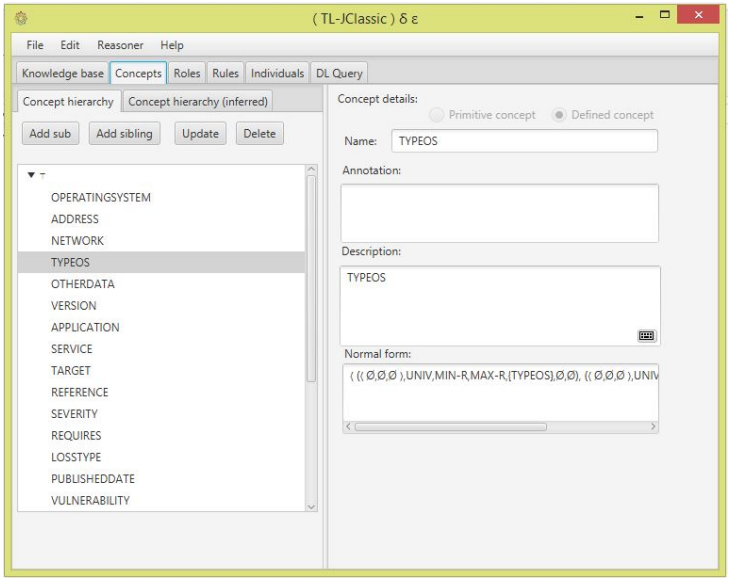

FIGURE 5.2 – Ensemble de concepts représentés par le raisonneur $T\_JClassic\delta\epsilon$

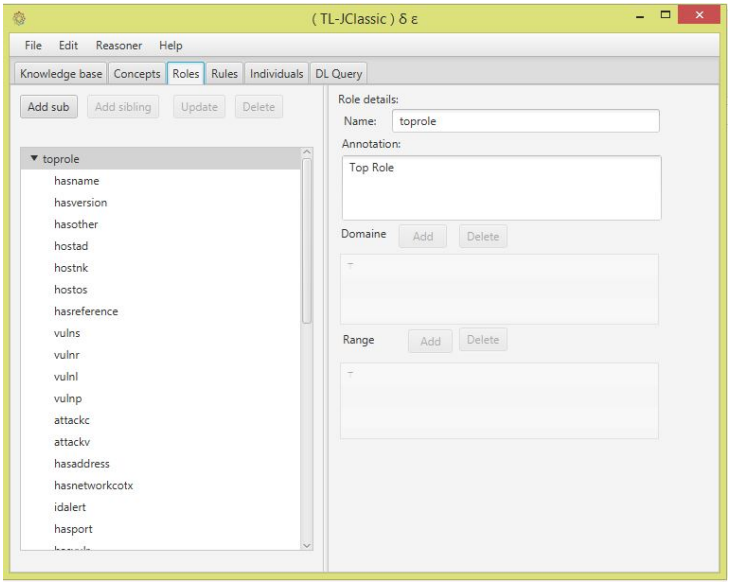

FIGURE 5.3 – Ensemble de rôles représentés par le raisonneur $T\_JClassic\delta\epsilon$

L'outil $T\_JClassic\delta\epsilon$ ne possède pas encore des plug-in qui permet de traiter des fichiers XML. Pour ces raisons, nous avons développé un ensemble de programmes afin de faire migrer les informations à partir de CVE et CAPEC comme des instances de ABOX.

5.3 Raisonnement et interrogation de l'ontologie

L'approche proposée repose sur certaines règles afin de déduire les alertes pertinentes en utilisant un ensemble d'informations contextuelles et les préférences de l'opérateur de sécurité.

Pour inférer de nouvelles connaissances, le raisonnement dans $JClassic\delta\epsilon$ s'appuie sur deux relations importantes, la relation de subsomption et la relation d'héritage. Le calcul de la relation de subsomption est utilisée pour classifier les concepts, quand la vérification de la subsomption entre deux concepts C et D consiste à trouver au moins une variable de mapping qui vérifie l'équivalence entre les concepts $C \sqcap D$ et C. Tandis que la relation d'héritage sert de base à la récupération des propriétés héritées d'un concept, en réalisant la soustraction des exceptions à la dénotation de concept C et le calcul des propriétés héritées pour ce concept.

Ces deux propriétés permettent au raisonneur $T\_JClassic\delta\epsilon$ d'appliquer l'ensemble des règles proposées afin de déduire les alertes pertinentes. Dans ce cas, l'arrivée de nouvelle alerte, le système vérifie la relation entre la vulnérabilité exploitée et le contexte de machine cibles.

5.4 Évaluation des systèmes de détection d'intrusions et la corrélation

L'évaluation des systèmes de détection d'intrusions permet d'améliorer les performances des IDS et savoir leur puissance et limite. Utiliser un bon benchmark représente un défi devant l'évaluation des systèmes de corrélation d'alerte. Cependant, la plupart des benchmarks publiés sont dédiés à l'évaluation des IDS et pas à l'évaluation des techniques de corrélation d'alerte, ainsi, ces benchmarks ne fournissent pas une description complète sur le réseau et sur le contexte des événements survenus.

Afin de démontrer l'efficacité de l'approche proposée, il était nécessaire d'utiliser une base d'attaque utile. Notre approche repose sur le contexte des entités du réseau et fournit une technique qui permet de traiter le changement de contexte en garantissant un raisonnement non monotone. La plupart des benchmarks publiés dans ce domaine ne fournissent pas de bonne description du contexte, et le plus important qu'ils ne traitent pas le changement de contexte. Pour ces raisons, nous avons généré une instance d'attaque qui répond aux besoins de notre expérimentation et qui permet d'évaluer l'efficacité de notre approche.

Dans cette section, nous allons présenter une étude sur certains benchmarks utilisé dans le domaine de la détection d'intrusion. Ensuite, nous décrivons la génération de l'instance d'attaque de type Island-hopping et les outils utilisés.

5.5 Les problèmes des benchmarks existants

Dans le domaine de la détection d'intrusion, où trouve certains benchmarks utilisés pour faire l'évaluation des approches proposées tels que CAIDA [CAI97], LBNL/ICSI Enterprise Tracing Project[LBN], DEF CON[DEF] ; KDD'99 datasets [KDD] et ISCX [SSTG12]. Dans cette section, nous présentons une description de chacune.

CAIDA recueille de nombreux types de données et les met à la disposition de la communauté des chercheurs. La plupart des ensembles de données de CAIDA sont très spécifiques à des événements ou à des attaques particulières. La plupart de ces traces sont des traces de squelette anonymisées avec leur charge utile, parfois leurs informations de protocole, leur destination, etc. sont complètement supprimées.

LBNL La recherche sur les ensembles de données de détection d'intrusion fait souvent référence à l'ensemble de données LBNL. La création du jeu de données LNBL était principalement motivée par l'analyse des caractéristiques du trafic réseau au sein des réseaux d'entreprises, plutôt que par la publication de données de détection d'intrusion. Selon ses créateurs, l'ensemble de données pourrait toujours être utilisé comme trafic d'arrière-plan pour les chercheurs en sécurité, car il contient un comportement

utilisateur presque exclusivement normal. L'ensemble de données n'est pas étiqueté, mais anonymisé pour des raisons de confidentialité et contient plus de 100 heures de trafic réseau au format paquet.

DEF CON DEF CON est une convention annuelle populaire des hackers. L'événement comprend une compétition de capture de flag (CTF : Capture The Flag) dans laquelle chaque équipe doit défendre son propre réseau contre les autres équipes tout en lançant des attaques simultanément sur les réseaux de ses adversaires. Le concours est généralement enregistré et disponible sous forme de paquet sur le site Web. Compte tenu de la nature de la concurrence, les données enregistrées contiennent presque exclusivement du trafic d'attaque et un comportement peu usuel de la part des utilisateurs. Le site Web est mis à jour chaque année avec les nouvelles données des compétitions de CTF

KDD Les jeux de données de détection d'intrusion KDD 99 sont basés sur l'initiative DARPA de 1998 visant à fournir aux concepteurs des systèmes de détection d'intrusions (IDS) un point de repère sur lequel évaluer différentes méthodologies.

Lincoln Labs a configuré un environnement permettant d'acquérir pendant neuf semaines des données de vidage TCP brutes pour un réseau local (LAN) simulant un réseau local LAN typique de l'armée de l'air américaine. Ils exploitaient le LAN comme s'il s'agissait d'un véritable environnement de la force aériennes, mais l'avaient multiplié d'attaques.

Les données brutes de formation contenaient environ quatre gigaoctets de données de vidage TCP binaires compressées provenant de sept semaines de trafic réseau. Cela a été traité dans environ cinq millions d'enregistrements de connexion. De même, les données de test de deux semaines ont généré environ deux millions d'enregistrements de connexion.

Une connexion est une séquence de paquets TCP commençant et se terminant à des heures bien définies, entre lesquelles des données circulent entre une adresse IP source et une adresse IP cible sous un protocole bien défini. Chaque connexion est étiquetée soit comme normale, soit comme une attaque, avec exactement un type d'attaque spécifique. Chaque enregistrement de connexion comprend environ 100 octets.

Les attaques se répartissent en quatre catégories principales :

DOS : déni de service, par exemple syn flood ; R2L : accès non autorisé à partir d'une machine distante, par ex. deviner le mot de passe ; U2R : accès non autorisé aux privilèges de super-utilisateur local (root), par exemple diverses attaques de type «buffer overflow» ; vérification : surveillance et autres procédures de détection, par exemple, balayage de port.

ISCX L'ensemble de données ISCX a été créé en 2012 [1] en capturant le trafic dans un

1. https ://www.unb.ca/cic/datasets/ids.html

environnement réseau émulé sur une semaine. Les auteurs ont utilisé une approche dynamique pour générer un ensemble de données de détection d'intrusion avec un comportement réseau normal et malveillant. Les soi-disant profils définissent des scénarios d'attaque tandis que les profils caractérisent le comportement normal de l'utilisateur, comme écrire des courriers électroniques ou naviguer sur le Web. Ces profils sont utilisés pour créer un nouvel ensemble de données dans un format basé sur un flux et un flux bidirectionnel. L'approche dynamique permet une génération continue de nouveaux ensembles de données. ISCX contient divers types d'attaques telles que SSH, DDS ou DDoS.

Les données benchmark constituent une bonne base pour évaluer et comparer la qualité des différents systèmes de détection d'intrusion sur le réseau. Donné un ensemble de données étiquetées dans lequel chaque point de données est attribué à la classe normale ou à l'attaque, le nombre d'attaques détectées ou le nombre de fausses alarmes peut être utilisé comme critère d'évaluation.

Malheureusement, il n'y a pas trop d'ensembles de données représentatifs. Selon Sommer et Paxson [SP10], le manque d'un ensemble de données représentative accessible au public constitue l'un des plus grands défis en matière de détection d'intrusion basé sur des anomalies. Des déclarations similaires sont faites par Malowidzki et al. [MBM15] et Haider et al. [HHS+17]. Cependant, la communauté travaille sur ce problème car plusieurs ensembles de données de détection d'intrusion ont été publiés au cours des dernières années.

Cet ensemble non exhaustif de benchmark est utilisé afin d'évaluer les travaux et les approches proposées dans le domaine de la détection d'intrusion selon leurs propres besoins. Bien que notre approche vise à traiter le changement de contexte, où on a besoin d'une description bien détaillée sur le contexte. Les benchmarks mentionnés ci-dessus ne fournissent pas une description complète sur l'environnement des réseaux lorsque les alertes sont générées. Ainsi, ces benchmarks décrivent le lancement des attaques sur certaines conditions fixées et ne traitent pas le changement du contexte courant. Pour ces raisons, nous somme pensés à lancer une instance d'attaque DDOS Island-hopping sous un contexte bien définit en changeant certaines conditions.

5.6 Description de l'attaque Island-hopping

L'attaque Island hopping décrit dans [SSTG12] comme le premier scénario de la base d'évaluation de la détection d'intrusion UNB ISCX. Island hopping est connue aussi sous le nom "leapfrogging" était autrefois connue comme une stratégie militaire dans laquelle les assaillants contretraient initialement leur stratégie sur des entités qui n'étaient pas leur cible d'origine mais qui pouvaient être exploitées pour atteindre la cible d'origine.

Islande hopping est également appliquée dans les attaques ciblées, où les attaquants appliquent la technique en évitant d'aller directement vers la cible. Au lieu de cela, les attaquants s'attaquent en premier lieu à des machines affiliées à leur cible (de préférence des machines qui ne sont pas nécessairement protégées). Les attaquants peuvent ensuite utiliser ces machines pour accéder à la cible. Les attaquants recherchent généralement d'autres systèmes connectés au système initialement compromis et tentent également de les pénétrer.

5.7 La génération d'une instance d'attaque Island-hopping

afin d'évaluer approche proposée dans le chapitre 4, nous présentons dans cette section les étapes suivis pour générer une attaque sous certaines conditions contextuelles en traitant leurs changements.

Nous avons lancé une instance de l'attaque Island-hopping, notre but lors de cette procédure est de lancer l'attaque sous un contexte bien défini, ensuite de changer le contexte (avoir de nouvelles vulnérabilités, fixer des vulnérabilités qui existent déjà, etc.) et capturer l'ensemble des nouvelles alertes.

Nous avons utilisé un réseau composé de deux LAN. LAN 1 avec l'adresse 10.0.5.0/24, LAN 2 avec l'adresse 10.0.2.0/24 avec la zone démilitarisée qui contient les serveurs, elle est identifiée par l'adresse 10.0.1.0/24, La figure 5.4 illustre la l'architecture de l'attaque survenue. Pour appliquer cette attaque, certaines machines exécutent une version vulnérable de l'application Adobe reader pour avoir un accès sur la machine et faire un balayage sur le réseau.

Le scénario de cette instance d'attaque est décrit dans le chapitre 6. Dans notre cas, l'attaquant génère le fichier malveillant en utilisant l'outil Metasploit en exploitant la vulnérabilité CVE-2008-2992, la machine attaquante dispose d'un port d'écoute sur lequel elle reçoit la connexion. Lors de l'ouverture d'une session entre l'attaquant et la cible, l'attaquant envoie un mail au nom d'admin@... avec un domaine similaire à une entreprise légitime à tous les utilisateurs du réseau afin d'améliorer la possibilité de trouver une machine vulnérable contenant un lien vers le fichier d'archives sur la machine de l'attaquant. Dans notre cas, lors de la création de du fichier PDF malicieux par Metasploit, l'accès au fichier PDF est fait par adresse IP pour éviter le blocage par mail (mesures de sécurité).

5.7.1 Génération du trafic réseau

Le trafic réseau est capturé par l'outil Wireshark qui il s'exécute sur les machines cibles durant le lancement de la séquence d'actions de l'attaque, il capture aussi les différentes communications entre les machines.

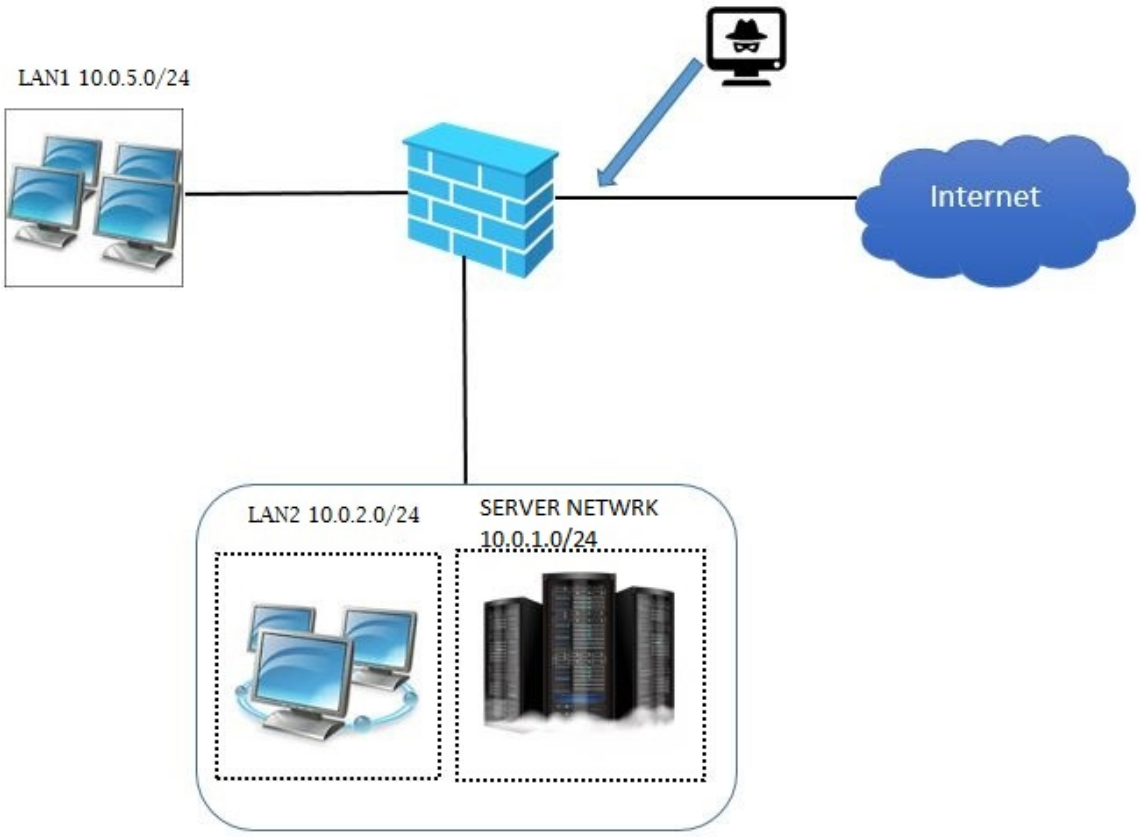

FIGURE 5.4 – L'architecture du réseau utilisée dans la génération d'une instance d'attaque de type Island-hopping

La capture du trafic facilite la procédure de test, le trafic est capturé sous format pcap qui est utilisé dans Tcpdump/Dump, snort et de nombreux autres outils réseau.

Afin de préparer les circonstances nécessaires pour changer le contexte, nous avons utilisé l'outil SIW (Advanced System Information for Windows) qui permet d'analyser les machines et recueillir des informations détaillées sur les propriétés et les paramètres du système (informations sur les logiciels, sur le matériel et sur le réseau et les outils) et les affiche de manière très compréhensible.

L'opération de capture du trafic réseau est réalisée pendant quelques heures en deux étapes, une fois pour le contexte initial et la deuxième pour le changement de contexte.

À la fin de cette procédure, nous avons analysé les trafics réseau par le système de détection d'intrusion Snort afin de générer l'ensemble des alertes et évaluer l'approche proposée.

5.8 Conclusion

Nous avons présenté deux parties dans ce chapitre. Dans la première, nous avons décrit les outils et la procédure d'implémentation de l'approche proposée, ainsi, nous avons exposé le raisonneur $T\_JClassic_{\delta\in}$ qui permet de répondre aux différentes requêtes et d'appliquer l'ensemble des règles proposées afin de déduire les alertes pertinentes.

Dans la deuxième partie, nous avons entamé l'évaluation des approches de corrélation d'alerte, nous avons présenté une étude sur les benchmarks les plus utilisés en abordant le problème de ne pas traiter le changement de contexte et le manque d'une description complète sur le contexte. Afin de remédier à ce problème, nous avons exposé dans ce chapitre les étapes et les outils utilisés pour créer un trafic réseau qui correspond une instance de l'attaque Island-hopping afin de collecter un ensemble d'alertes générées dans un contexte bien défini en traitant le changement de contexte.

Dans le chapitre suivant, nous allons présenter nos expérimentations en discutant leur résultats.

Chapitre 6

Expérimentation et résulats

6.1 Introduction

L'approche proposée dans le chapitre 4 vise à réduire le nombre d'alertes non pertinentes, en se basant sur la représentation du contexte et le changement qui peut se produire.

Afin de démontrer l'efficacité de cette approche, nous l'avons expérimenté sur le premier scénario LLDDOS 1.0 du jeu de données DARPA 2000. Ainsi, pour bien évaluer le traitement du contexte et le changement du contexte, nous avons expérimenté l'approche NOC-IDS sur une instance de l'attaque Island-hopping généré dans un contexte bien définit.

Dans ce chapitre, nous allons présenter deux cas d'étude en deux section avec la discussion de leurs résultats et les comparer avec d'autre travaux dans ce domaine.

6.2 Cas d'étude : instance de Island-hopping

Dans cette instance (voir la figure 6.1), l'attaquant crée un fichier pdf malicieux et l'envoie pour pouvoir ouvrir une session avec un shell inversé TCP sur le port 5555. L'attaquant réussi à ouvrir une session avec la machine de l'adresse IP 10.0.5.168 qui satisfait la vulnérabilité CVE-2008-2992 [1]. En utilisant cette machine, l'attaquant lance un scan NMAP sur les machines connectées au réseau 10.0.2.0/24.

L'utilisateur de l'adresse IP 10.0.2.235 est identifié, ce dernier exécute un Windows XP sp1 avec un protocole d'authentification SMB vulnérable sur le port 445. L'attaquant exploite la vulnérabilité CVE-2008-4037 pour prendre un accès priviligié sur la machine correspondante. Un scan NMAP est effectué depuis la machine de l'adresse 10.0.2.235 sur le réseau des serveurs.

1. Stack-based buffer overflow dans Adobe Acrobat Reader 8.1.2 et les versions antérieures permet aux attaquants distants d'exécuter du code arbitraire via un fichier PDF qui appelle la fonction JavaScript util.printf avec un argument de chaîne de format spécialement conçu

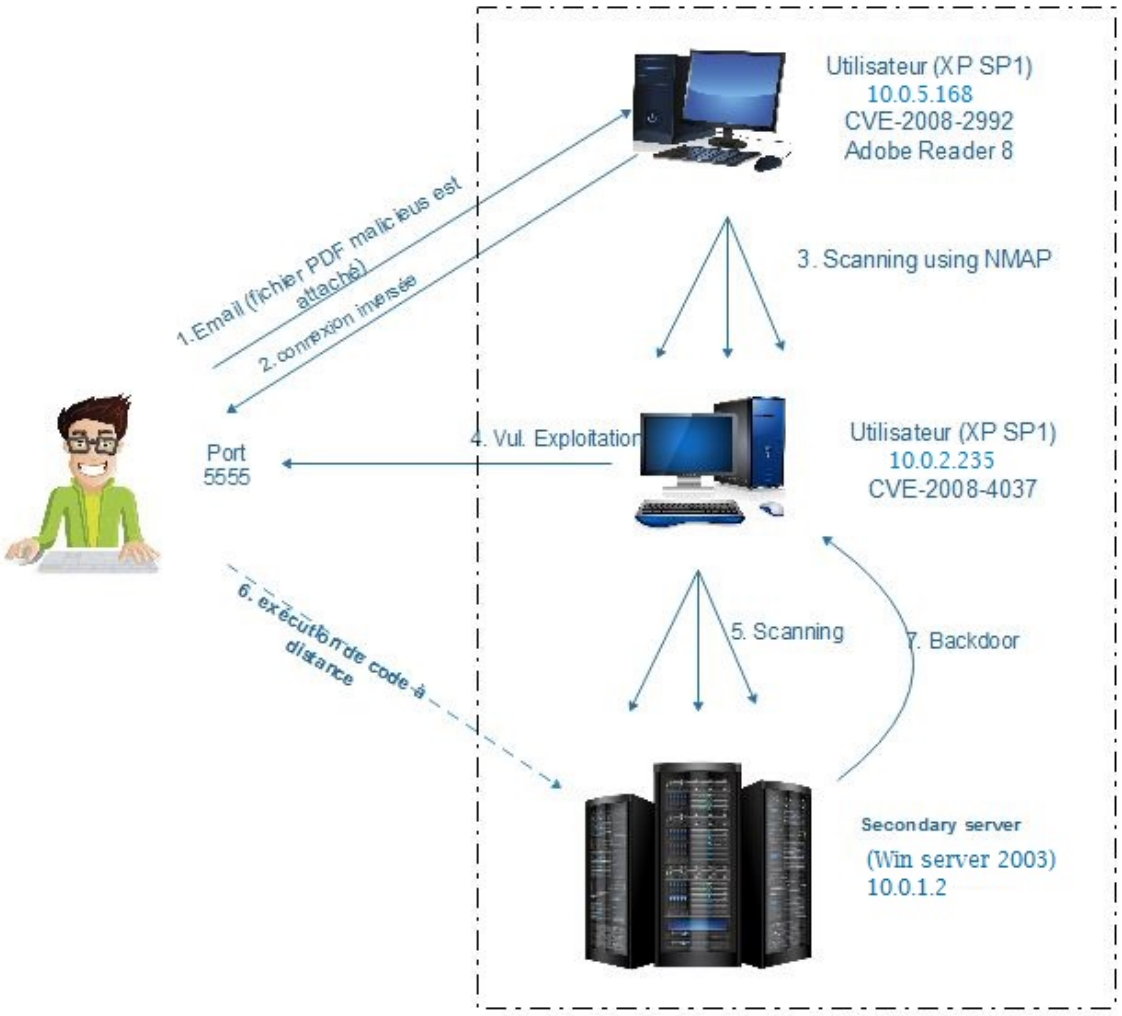

FIGURE 6.1 – Le scénario d'attaque Island-hopping généré

L'attaquant identifie un Windows Server 2003 actif qui exécute une application Web interne utilisant My SQl server. Cela conduit l'attaquant à effectuer une attaque par injection SQL. L'attaquant utilise cette vulnérabilité pour compromettre le serveur cible.

Afin d'appliquer notre système de corrélation, les informations contextuelles et les vulnérabilités sont collectées et stockées dans des fichiers XML. Le tableau 6.1 représente les machine (10.0.5.168, 10.0.2.235, 10.0.1.2), l'environnement de chacune et les vulnérabilités qui leur affectes.

TABLE 6.1 – Le contexte et les vulnérabilités pour chaque machine

Adresse IP	port ouvert	services	Application	Vulnérabilité
10.0.5.168	5555	-	Adobe reader8.1	CVE-2008-2992
10.0.2.235	-	-	Windoxs xp	CVE-2008-4037
10.0.1.2	80	My SQL server Application web	Windows server 2003	-

Nous utilisons le trafic généré dans le contexte actuel afin de générer les alertes correspondantes. Premièrement, on applique la règle *hasApplication* pour décrire les machines qui exécutent l'application Adobe reader version 8.1.

hasApplication (Adobe_reader) $\sqsubseteq \forall$ *AppName :* $\{$*Adobe Reader}* $\sqcap$ *hasAddress :* $\{$*10.0.5.168}.*

La vulnerability buffer overflow est définit comme suit :

Vulnerability(Adobe util.printf) $\sqsubseteq \forall$ *hasReference : CVE-2008-2992* $\sqcap$ *hasReference AT-LEAST 1* $\sqcap\forall$ *VulnS :high* $\sqcap$ *VulnS FILLS* $\{$*low,medium,high*$\}$ $\sqcap\forall$ *VulnR :remote* $\sqcap$ *VulnR FILLS* $\{$*remote, local, user*$\}$ $\sqcap\forall$ *VulnL : privilege_escalation*$\sqcap$ *VulnL FILLS* $\{$*confidentiality, integrity, availability, privilege_escalation*$\}$ $\sqcap\forall$ *VulnP :2008* $\sqcap$ *VulnP AT-LEAST 1* $\sqcap$ *vulnContext : OperatingSystem (XP).*

La règle δ *VulnerableA* vise à déduire les machine qui sont affectées par la vulnérabilité buffer overflow par défaut.

δ *VulnerableA (buffer_overflow)* $\sqsubseteq$ *VulAPP :hasApplication*$\sqcap$ *VulAPP :* $\{$*Adobe reader}* $\sqcap\forall$ *VulAT :Target* $\sqcap$ *VulAT :* $\{$*PC1}* $\sqcap\forall$ *VulnAVulnerability : Vulnerability* $\sqcap$ *VulnAVulnerability : Adobe util.printf.*

Nous définissons la représentation logique du système d'exploitation cible comme suit :

OperatingSystem (XP)$\sqsubseteq \forall$ *hasType :Windows* $\sqcap$ *hasType AT-LEAST 1* $\sqcap$ $\forall$ *hasVersion :XP* $\sqcap$ *hasVersion AT-LEAST 1*$\sqcap\forall$ *hasOData :SP1*

La règle suivante décrit les machines cibles qui exécutent Windows XP comme système d'exploitation.

Target (user-lan2)$\sqsubseteq$ *HostAd :* $\{$*10.0.2.235}* $\sqcap$ *HostNK :* $\{$*LAN2}* $\sqcap$ *HostOs :XP*

Afin de déduire quel hôte est affecté par défaut par la vulnérabilité du protocole d'authentification SMB référencée par CVE-2008-4037, notre modèle applique la règle suivante. δ *VulnerableS (V1)*$\sqsubseteq$ *VulSTarget :* $\{$*user-lan2}* $\sqcap$ *VulSService :* $\{$*remote SMB}* $\sqcap$ *VulSVulnarebility :* $\{$*CVE-2008-4037}* .

Afin de répondre aux requêtes de pertinence, nous utilisons les règles δ *relevantAlert3* et δ *relevantAlert2*

δ *relevantAlert2* $\sqsubseteq \forall$ *IdAlert :ID* $\sqcap$ *IdAlert AT-LEAST 12*$\sqcap\forall$ *RlvAlrtA :Rverse connexion* $\sqcap$ $\sqcap$ *RlvAlrtT : Target* $\sqcap$ *RlvAlrtT :* $\{$*user-lan2* $\}$ $\sqcap\delta$*VulnerableS(V1)*˙

δ *relevantAlert3 (RA1)*$\sqsubseteq \forall$ *IdAlert :ID* $\sqcap$ *IdAlert AT-LEAST 1*$\sqcap\forall$ *RlvAlrtA :Attack* $\sqcap$ *RlvAlrtA : Reverce_connexion* $\sqcap$ *RlvAlrtT : Target* $\sqcap$ *RlvAlrtT :* $\{$*user-lan1* $\}$ $\sqcap\delta$ *VulnerableA (buffer-overflow).*

La règle δ *relevantAlert3* signifie que l'alerte est pertinente si elle est générée à la suite d'une attaque qui tente de compromettre l'application Adobe reader. Cette attaque exploite la vulnérabilité buffer overflow et la machine cible user—lan1 exécute cette application dans laquelle elle est vulnérable par défaut.

La règle δ relevantAlert2 signifie que l'alerte est pertinente si elle est générée à la suite d'une attaque qui tente de compromettre la sécurité de la machine user-lan2 qui est par défaut vulnérable. Pour déduire la pertinence des alertes le système applique la règle *IS_ relevantAlert.*
IS_ relevantAlert $\sqsubseteq \delta$ *relevantAlert3(RA1).*

Afin de traiter le cas de changement du contexte, nous avons laisser la vulnérabilité (buffer overflow) et nous avons changé le système d'exploitation d'user-lan2. Pour répondre à une autre demande de pertinence où la vulnérabilité bufferoverflow est fixée, nous définissons l'exception d'être vulnérable.

On peut définir δ *VulnerableA(BO)* comme une exception de *VulnerableA(buffer-overflow)* par défaut :
(δ VulnerableA(BO))$^{\epsilon} \sqsubseteq \delta$ VulnerableA(CVE-2008-2992).

Dans $JClassic^{\epsilon}_{\delta}$, l'exception est subsumée par l'exception du fait par défaut ($(\delta A)^{\epsilon} \sqsubseteq A^{\epsilon}$). Donc, *VulnerableA(CVE-2008-2992)*$^{\epsilon} \sqsubseteq \delta$ *VulnerableA(BO).*

Qui fait la définition de δ relevantAlert3 (RA2) comme suit :
(VulnerableA(CVE-2008-2992))$^{\epsilon} \sqsubseteq \delta$ *relevantAlert3 (RA2).*
et qui pousse à ne pas déduire Is_RelevantAlert
Is_ RelevantAlert $\not\sqsubseteq$ *(VulnerableA(CVE-2008-2992))*$^{\epsilon}$.

En conséquence, l'alerte générée dans ce nouveau contexte n'a pas pu être sélectionnée comme pertinente. Le tableau **??**, résume la base des individus ABOX utilisée dans cette cas d'étude.

Address *(10.0.2.235)*, ***Address****, (10.0.5.168)*, ***Address*** *(10.0.1.2)*
String *(adobe reader)*, ***String*** *(authentification. (SMB) (LAN1)*,
Network *(LAN1)*, ***Network*** *(LAN2)*, ***Network*** *(server)*
TypeOS *(Windows)*
Version*(XP)*
Version*(Server2008)*
Reference *(CVE-2008-2992)*, ***Reference*** *(CVE-2008-4037)*
Severity *(low)* , ***Severity*** *(medium)*, ***Severity*** *(high)*
Requires *(remote)*, ***Requires*** *(local)*, ***Requires*** *(use)*
LossType *(confidentiality)*, ***LossType*** *(integrity)*, ***LossType*** *(availability)*,
LossType *(privilege_ escalation)*
PublishedDate *(2008)*

port(445), ***port(5555)***, ***Classification*** *(icmp_ping)*
protocole (ICMP) protocole (IP), *protocole (TCP)*, *protocole (UDP)*,
Vulnerability *(remote_SMB)* ***hasApplication*** *(Adobe_reader)*
OperatingSystem (XP), ***OperatingSystem*** *(XPsp1)*,
OperatingSystem *(WinServer2008)*, ***Target(PC1)***, ***Target*** *(user-lan2)*,
Attack*(Reverse_connexion)*, ***AttackC****(Reverse_connexion, BufferOverFlow)* ,
AttackV *(Reverse_connexion, Adobe util.printf)* , ***Attack****(Vul_exploitation)* ,
AttackC *(Vul_exploitation, icmp)* ***AttackV****(Vul_exploitation, Remote SMB)* ,
hasType*(XP,Windows)*, ***hasType****(xpSp1,Windows)*, ***hasType****(WinServer2008,Windows)*,
hasVersion*(XP,xp)*, ***hasVersion****(xpSp1,xp3)*, ***hasVersion****(winServer2008,Windows server 2008)* ,
OtherData*(XP,sp3)*, ***OtherData****(xpSp1, sp1)*, ***HostAd****(user-lan2,10.0.2.235)*,
HostNK*(user-lan2, LAN2)*, ***HostOS****(user-lan, XP)*,
hasReference*(Adobe uril.printf, CVE-2008-2992)*, ***VulnS****(Adobe uril.printf, high)*,
VulnR*(Adobe uril.printf, remote)*, ***VulnP****(Adobe uril.printf, 2008)*,
VulnContext(Adobe uril.printf, XP),
hasReference*(remote SMB, CVE-2008-4037)*, ***VulnS****(remote SMB, high)*,
VulnR*(remote SMB, remote)*, ***VulnS****(remote SMB, high)*, ***VulnR****(remote SMB, remote)*,
VulnL*(remote SMB, privilege_escalation)*, ***VulnP****(remote SMB, 2008)*,
VulnContext*(remote SMB, xpSp1)*, ***AttackC****(Reverse_connexion, bufferOverFlow)*,
AttackV*(Reverse_connexion, Adobe util.printf)*,
AttackC*(vul_exploitation, icmp_ping)*, ***AttackV****(remoteSMB, Adobe util.printf)*,
δ VulnerableS (V1), ***VulSTarget(V1, user-lan2)***,
VulSService(V1, AuthantificationSMB), ***AppName(Adobe_reader, Adobe reader)***,
hasAdress(Adobe_reader, 10.0.5.168), ***VulAPP(buffer_overflow, Adobe_reder)***,
VulAT(buffer_overflow, PC1), ***VulnAVulnerability(buffer_overflow, Adobe util.printf)***,
AppName(Adobe_reader, Adobe reader), ***hasService(AuthantificationSMB)***,
servicePort(AuthantificationSMB, 445),
serviceVulR(AuthantificationSMB,CVE-2008-4037), ***hasAdress(AuthantificationSM***
10.0.2.235),

TABLE 6.2 – Résumé de la base d'assertions ABOX utilisée dans le cas d'étude island-hopping

Le capture du trafic généré est fait sur plusieurs hôtes qui résulte plusieurs fichiers pcap. Lors de lancement de l'attaque et pour mimer un cas réal dans un réseau, nous avons lancé certaines activités qui peuvent ce produisent dans un réseau où les utilisateurs envoient des mails, des ping, download des fichiers, etc. En conclusion, l'analyses des fichiers pcap par Snort résulte un total de 171 alertes avec une quantité d'alertes redondantes et de

type ICMP.

le système d'inférence déduit 27 alerte pertinente, Le tableau 6.3 illustres la correspondance entre les actions de cette instance d'attaque et les alertes détectées. depuis un total de 171 alertes.

TABLE 6.3 – Actions réalisées dans l'attaque island hopping

Nbr d'alertes	Action	Type d'alerte	Source	Destination
2	Configurer un Meterpreter reverse TCP shell	Local Exploit	10.0.5.168	10.0.5.168
1	l'accès au PDF malicieux	Reverse connexion	10.0.5.168	10.0.5.168
16	NMAP scan	scanning	10.0.5.168	10.0.2.0/24 10.0.5.0/24
2	Identifier l'hôte actif avec la vulnérabilité SMB	Windows File Sharing	10.0.5.168	10.0.2.235
1	exploiter la vulnérabilité (CVE-2008-4037)	Local exploit	10.0.2.235	10.0.2.235
2	NMAP scan depuis l'utilisateur compromis	Scanning alert	10.0.2.235	10.0.1.0/24
1	Découvrir une application web interne	HTTP web	10.0.2.235	10.0.1.2
1	Lancer une attaque injection SQL	SQL injection	10.0.1.2	10.0.1.2
1	Exécuter des Bad_request	bad request	10.0.1.2	10.0.1.2

Après de monter la version de Windows xp sp1 à une version plus récente qui implique un changement de contexte. Le système détecte 19 alerte pertinente.

Le nombre d'alertes pertinentes est réduit à cause de l'exception qui a été faite sur la règle δ *Is_ Relevant* après le changement de la vulnérabilité CVE-2008-4037.

6.3 Cas d'étude : DARPA 2000 LLDDOS1.0

Afin de démontrer l'efficacité du modèle proposé, nous avons utilisé l'ensemble de données DARPA 2000 (Défense Advanced Research Projets)en raison de la documentation décrivant les attaques préparées. DARPA 2000 est parrainée par le laboratoire de la recherche de la force aérienne, et gérée par le laboratoire Lincoln du MIT.

Cet ensemble de données est une trace bien étudiée, documentée et accessible au public dans le domaine de la détection d'intrusion. L'ensemble de données DARPA 2000 contient deux scénarios d'attaque du type déni de service distribué (DDOS). Nous sommes intéressés par le scénario LLDDOS 1.0 (Scénario de laboratoire Lincoln (DDoS) 1.0) pour démontrer la capacité de l'approche NOC-IDS à identifier les alertes pertinentes.

L'attaque de LLDDOS 1.0 est répartie sur cinq phases, comme il est illustré dans le tableau 6.4, dans lequel l'attaque exploite la vulnérabilité Remote-To-Root Sadmind de l'outil Sadmind pour obtenir l'accès root sur les trois hôtes Salaris du site de la base aérienne et lancer une attaque par déni de service distribué (DDoS) sur le site du gouvernement américain. Par conséquent et tout d'abord, l'attaquant analyse le réseau pour déterminer quels hôtes sont "actifs", puis utilise l'option ping du programme exploité Sadmind pour vérifier l'existence du service Sadmind sur les hôtes découverts.

Le script d'attaque tente d'exploiter la vulnérabilité Remote-to-Root Sadmind plusieurs fois contre chaque hôte et à chaque fois avec des différents paramètres pour pénétrer dans des machines vulnérables. Après cela, l'intrus a réussi à compromettre les trois hôtes (Mill, Pascal, Locke) qui fournissent le service Sadmind avec les adresses IP 172.16.115.20, 172.16.112.50, 172.16.112.10. L'attaquant utilise *telnet, rcp et rsh* pour installer un programme DDos sur les hôtes compromis.

Table 6.4 – Phases du scénario d'attaque LLDDOS 1.0

Phase	Nom	Temps	Description
1	Balayage IP	09 :45 à 09 :52	L'attaquant envoie des requêtes d'écho ICMP pour savoir quels hôtes sont actifs
2	Sadmind Ping	10 :08 à 10 :18	Tester l'existence de sadmind démon sur les IP en direct
3	Exploiter	10 :33 à 10 :34	Exploiter la vulnérabilité sadmind percer des hôtes vulnérables
4	Installation	10 :50	Installation du cheval de Troie mstream Logiciel DDoS sur trois hôtes
5	Lancement DDoS	11 :27	Lancer l'attaque DDoS

La relecture des données "Inside-tcpdump" du scénario LLDDOS1.0 conduit à générer une grande quantité d'alertes par le système de détection d'intrusion Snort [2] IDS. Ce grand nombre d'alertes est généré en raison de nombreuses tentatives d'intrusion pour atteindre les hôtes vulnérables.

Afin d'appliquer les règles de l'approche proposée NOC-IDS en représentant les données LLDDOS 1.0, nous définissons d'abord manuellement le contexte des hôtes existant dans la documentation DARPA 2000. La même chose est faite pour définir les vulnérabilités, y compris les différentes fonctionnalités de la base de données en ligne NVD et CVE.

2. Snort est un système de détection d'intrusion ouvert, http ://www.snort.org

Ensuite, nous appliquons l'ensemble des règles et des concepts NOC-IDS, comme nous les expliquerons au cours de ce chapitre.

Pour mettre en évidence le problème, nous avons d'abord ajouté à la base de connaissances ABOX les trois hôtes (Mill, Pascal, Locke) qui fournissent le service Sadmind. Pour cela, nous appliquons le concept Target qui décrit son réseau et son système d'exploitation de la manière suivante :

Target (mill)$\sqsubseteq$*HostAd :* {*172.16.115.20*} $\sqcap$ *HostNK :* {*Air Force Base*} $\sqcap$ *hasType :* {*Windows NT*}

Ce concept indique que l'hôte nommé "mill" et identifié par l'adresse 172.16.115.20, exécute Windows NT en tant que système d'exploitation et qu'il appartient au réseau de la base aérienne.

Target (pascal)$\sqsubseteq$ *HostAd :* {*172.16.112.50*} $\sqcap$ *HostNK :* {*Air Force Base*} $\sqcap$ *hasType :* {*Windows NT*}

Ce concept indique que l'hôte nommé "pascal" et identifié par l'adresse 172.16.112.50, exécute Windows NT en tant que système d'exploitation et qu'il appartient au réseau de la base aérienne.

Target (Locker)$\sqsubseteq$ *HostAd :* {*172.16.112.10*} $\sqcap$*HostNK :* {*Air Force Base*} $\sqcap$ *hasType :* {*Windows NT*}.

Ce concept indique que l'hôte nommé "Locker" identifié par l'adresse 172.16.112.10, exécute Windows NT en tant que système d'exploitation et qu'il appartient au réseau de la base aérienne.

Ensuite, nous appliquons la règle *hasService* pour décrire quel hôte fournit le service Sadmind avec la vulnérabilité référencée *CVE-1999-0977*.

hasService (Sadmind) $\sqsubseteq$ *serviceName :* {*Sadmind*} $\sqcap$ *hasAddress {172.16.115.20}* $\sqcap$ servicePort FILLS { Telnet, rsh, rpc } $\sqcap$ $\forall$ serviceVulR :Reference $\sqcap$ serviceVulR : {CVE-1999-0977}.

hasService (Sadmind) $\sqsubseteq$ *serviceName :* {*Sadmind*} $\sqcap$ *hasAddress {172.16.112.50}* $\sqcap$ servicePort FILLS { Telnet, rsh, rpc } $\sqcap$ $\forall$ serviceVulR :Reference $\sqcap$ serviceVulR : {CVE-1999-0977}.

hasService (Sadmind) ⊑ *serviceName :* {*Sadmind*} ⊓ *hasAddress {172.16.112.10}* ⊓ servicePort FILLS { Telnet, rsh, rpc } ⊓ ∀ serviceVulR :Reference ⊓ serviceVulR : {CVE-1999-0977}.

Les résultats de ces règles indiquent que les trois hôtes dotés de l'adresse IP 172.16.115.20, 172.16.112.50, 172.16.112.10 fournissent le service Sadmind entendu sur les ports telnet, rsh et rpc, ce service a pour référence la vulnérabilité *CVE-1999-0977.*, Afin de spécifier quel hôte du réseau est vulnérable avec la vulnérabilité référencée *CVE-1999-0977.*

Cette vulnérabilité est définit comme suit :

Nous appliquons la règle vulnérable par défautδ *VulnerableS* afin de déduire les machines qui satisfont le contexte sous-jacent par défaut comme suit :

δ VulnerableS (VS1)⊑ VulSTarget FILLS {mill, pascal,Locker} ⊓ VulSService : {Sadmind}⊓ VulSVulnarebility : {CVE-1999-0977} .

En analysant les alertes générées pour les données "Inside-tcpdump" du scénario LLDDOS1.0, nous observons différents types d'actions qui ont été lancées pour réaliser une attaque buffer-overflow. Ces actions sont illustrées dans le tableau 6.5.

Table 6.5 – Actions Buffer-overflow dans le scénario LLDDOS1.0

Action	Description
A1 : icmp_ping	demandes d'écho ICMP dans ce balayage et écoute echo-replies pour déterminer quels hôtes sont "active"
A2 : rpc_sadmind_request	Demande RPC (Remote Procedure Call) pour le service sadmind
A3 : sadmind_ping	demande de vérification de l'existence de Sadmind
A4 : sadmind_root_query	demande au serveur Sadmind avec privilège d'administrateur
A5 : sadmind_bof	tentatives d'attaque par buffer overflow sur Sadmind
A6 : icmp_reply	ICMP-Reply pour confirmer que l'hôte soit "active"
A7 : telnet_info	établir une session Telnet avec succès
A8 : telnet_login_incorrect	tente d'établir Telnet session avec échec
A9 : telnet_bad_login	tente d'établir Telnet session avec échec
A10 : rsh_root	établir une session RSH (shell distant) avec succès
A11 : icmp_port_unreachable	réponse négative sur ping (la machine n'existe pas)

Nous sommes intéressés pour représenter l'attaque buffer-overflow avec un ensemble d'actions illustrées dans le tableau 6.5, nous définissons l'attaque buffer-overflow avec la

vulnérabilité d'exploitation de la manière suivante :
Attack (Buffer-Overflow)$\sqsubseteq$*AttackC FILLS* $\{$*icmp_ping, rpc_sadmind_request, sadmind_ping, sadmind_root_query, sadmind_bof* $\}$ $\sqcap$ *AttackV : Solaris_sadmind.*

Le modèle proposé peut donner une réponse à chaque demande pour déterminer la pertinence des alertes. pour répondre à cette demande, nous appelons la règle δ *relevantAlert2* :
δ *relevantAlert2 (RA1)* $\sqsubseteq$ $\forall$ *IdAlert :ID* $\sqcap$ *IdAlert AT-LEAST 1* $\sqcap$ *RlvAlrtA :* $\{$*Buffer-Overflow*$\}$ $\sqcap$ *RlvAlrtT FILLS* $\{$*mill, pascal, Locker*$\}$ $\sqcap$ δ *VulnerableS(VS1).*
On peut alors déduire la pertinence d'alerte en appliquant la règle *Is _relevantAlert*
Is _relevantAlert $\sqsubseteq$ δ *relevantAlert2(RA1)*

Pour répondre à une autre demande de pertinence lorsque le contexte a été modifié, c'est-à-dire si une autre hôte cible ne satisfait pas le contexte sous-jacent (la machine ne fournit pas le service Sadmind ou ne remplit pas les conditions nécessaires pour être affectée par la vulnérabilité de serveur Sadmind), nous définissons l'exception d'être vulnérable. Nous pouvons définir δ VulnerableS (VS2) comme une exception de la vulnérabilité par défaut (VS1) comme suit :
(δ *VulnerableS(VS1))*$^{\epsilon}$ $\sqsubseteq$ *(*δ *VulnerableS(VS2)).*

Dans $JClassic^{\epsilon}_{\delta}$, l'exception est subsumée par l'exception du fait par défaut ($(\delta A)^{\epsilon} \sqsubseteq A^{\epsilon}$). Ainsi, VulnerableS (VS1) $^{\epsilon}$ $\sqsubseteq$ δ VulnerableS (VS2) .

Lesquels font la définition de δ relevantAlert2 (RA2) comme suit :
(VulnerableS (VS1)) $^{\epsilon}$ $\sqsubseteq$ δ relevantAlert2 (RA2)
et cela conduit à ne pas déduire *Is_ RelevantAlert*
Is_ RelevantAlert $\not\sqsubseteq$ *(VulnerableS (VS1))*$^{\epsilon}$.

En conséquence, l'alerte générée dans ce nouveau contexte n'a pas pu être sélectionnée comme pertinente. Le tableau 6.6 résume la base des individus utilisé dans cette cas d'étude.

6.3.1 Résultats et Discussion

L'application des règles précédentes sur les données "Inside _tcpdump" du scénario LLDDOS1.0 dans le fichier DARPA 2000 permet de déduire uniquement les alertes pertinentes générées pour l'ensemble des machines cibles qui satisfont le contexte de la vulnérabilité du service Sadmind d'exploitation.

La relecture des données du premier scénario LLDDOS 1.0, implique la génération d'un total d'alertes de 33874. Le trafic capturé correspond au scénario survenu pendant trois heurs sur les réseaux 172.16.115.0/24, 172.16.114.0/24, 172.16.113.0/24, 172.16.112.0/24.

ABOX
Address *(172.16.115.20)*, ***Address***, *(172.16.112.10)*, ***Address*** *(10.0.1.2)*
Network *(Air Force Base)*,
String*(sadmind)*, ***TypeOS*** *(Windows)*,
Version*(NT)*,
port*(rpc)*, ***port****(telnet)*, ***port****(rsh)*, ***Reference*** *(CVE-1999-0977)* ,
Severity *(low)* , ***Severity*** *(medium)*, ***Severity*** *(high)*
Requires *(remote)*, ***Requires*** *(local)*, ***Requires*** *(use)*,
LossType *(confidentiality)*, ***LossType*** *(integrity)*, ***LossType*** *(availability)*,
LossType *(privilege_ escalation)*,
Vulnerability*(Solaris_ sadmind)*, ***PublishedDate*** *(1999)*,
Classification *(icmp_ ping)*, ***Classification*** *(rpc_ sadmind_ request)*,
Classification *(sadmind_ ping)*, ***Classification*** *(sadmind_ root_ query)*,
Classification *(sadmind_ bof)*
Service *(Sadmind)*,
OperatingSystem *(Windows NT)*, ***hasType*** *(OperatingSystem, Windows)*,
hasVersion*(OperatingSystem, NT*, ***Target*** *(mill)*, ***HostAd*** *(mill,172.16.115.20)*,
HostNK *(mill, Air Force Base)*, *hasType(mill, Windows NT)*,
Target *(pascal)* ,***HostAd*** *(pascal, 172.16.112.50)*, ***HostNK****(pascal,Air Force Base)*,
hasOS *(pascal, Windows NT)*, ***Vulnerability****(Solaris_ sadmind)*,
hasReference *(Solaris_ sadmind, CVE-2008-2992)*,
VulnS*(Solaris_ sadmind, high)*,
VulnR *(Solaris_ sadmind, remote)*, ***VulnL****(Solaris_ sadmind, privilege_ escalation)*,
VulnP*(Solaris_ sadmind, 1999)*, ***vulnContext*** *(Solaris_ sadmind, Windows NT)*
Attack *(Buffer-Overflow)*, ***AttackC****(Buffer-Overflow, sadmind_ ping)*,
AttackV*(Buffer-Overflow, Solaris_ sadmind)*
hasService*(Sadmind)*, ***serviceName****(Sadmind, sadmind)*,
hasAddress*(Sadmind, 172.16.115.20)*, ***servicePort****(Sadmind,rpc)*,
serviceVulR*(Sadmind, CVE-1999-0977)*, ***hasAddress****(Sadmind, 172.16.112.50)*,
hasAddress*(Sadmind, 172.16.112.20)*, δ ***VulnerableS****(VS1)*,
VulTarget*(VS1, mill)*, ***VulTarget****(VS1, pascal)*, ***VulTarget****(VS1, locker)*,
Atack*(Buffer-overflow)*, ***AttackC****(Buffer-overflow,icmp_ ping)*,
AttackC*(Buffer-overflow,rpc_ sadmind_ request)*, ***AttackC****(Buffer-overflow,icmp_ ping)*,
AttackC*(Buffer-overflow,sadmind_ ping_ sadmind)*,
AttackC*(Buffer-overflow,sadmind_ root_ query)*,
AttackC*(Buffer-overflow,sadmind_ bof)*,
AttackV*(Buffer-overflow,Solaris_ sadmind)*,

TABLE 6.6 – Résumé de la base d'assertions ABOX utilisée dans le cas d'étude DARPA 2000

l'application de l'ensemble des règles d'inférence d l'approche NOC-IDS avec le contexte montré dans tableau 6.7, fait la déduction de 93 alertes comme pertinentes dans les cinq étapes du scénario. Les alertes déduites correspondent à la réussite des requêtes de l'attaquant à compromettre les machines qui satisfont la vulnérabilité exploitée. Selon le principe de notre approche, lorsque la destination ne satisfait pas la vulnérabilité exploitée, l'alerte ne se présente pas à l'opérateur de sécurité en premier lieu. Le reste des alertes générées

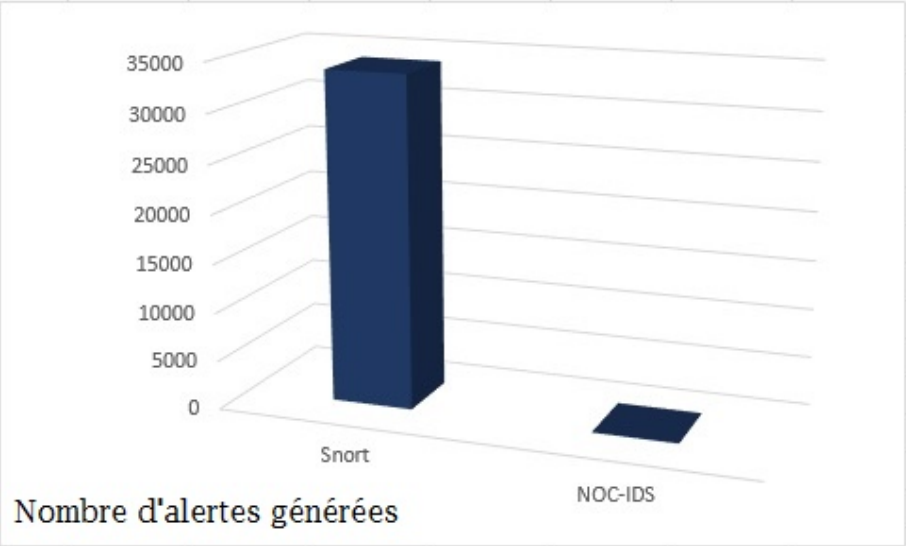

FIGURE 6.2 – Le nombre d'alertes générées par Snort et celles déduites par le modèle proposé comme pertinentes

par Snort correspondent aux requêtes où la cible ne se présente pas dans les conditions des vulnérabilités exploitées.

DARPA

TABLE 6.7 – Le contexte des machines cibles dans le scénario LLDDOS1.0

Mill	Pascal	Lock
172.16.115.20 Service sadmid rcp Windows NT	172.16.112.50 service Sadmind rcp Windows NT	172.16.112.10 Service Sadmind rcp Windows NT

L'approche NOC-IDS permet d'analyser les alertes correspondantes aux intérêts des opérateurs de sécurité. Il considère les alertes générées pour les attaques ayant échoué comme de faux positifs. Par exemple les alertes qui sont générées dans la deuxième étape de l'attaque et qui correspondent à des machines qui n'exécutent pas le service sadmind (voir la figure 6.3). La réponse sur les requêtes rcp en prend comme destination l'adresse de l'attaquant par icmp_destination_unreachable, cet ensemble d'alertes ne sont pas parmi les alertes pertinentes déduite par la règle δ InNetwork.

Ces principes peuvent interpréter les résultats présentés dans la figure Les figures 6.2 représente la déférence entre le nombre d'alertes qui peut être analysé par l'IDS Snort et le nombre d'alerte qui doit être analysé à l'utilisation de ce système d'inférence.

Les résultats obtenus indiquent l'efficacité de l'approche proposée à réduire le nombre d'alertes non pertinentes et de faux positifs.

Le modèle proposé fonctionne comme un outil complémentaire au système de détection d'intrusion, dans lequel les règles d'inférence proposées sont appliquées à l'ensemble des

```
<IDMEF-Message version="0.1">
<Alert alertid="28" impact="unknown" version="1">
<Time>  <date>03/07/2000</date> <time>10:15:10</time>
<sessionduration>00:00:55</sessionduration></Time>
<Analyzer ident="tcpdump_inside">
<name>tcpdump_inside</name></Analyzer>
<Source spoofed="unknown"><Node><Address category="ipv4-addr">
<address>172.16.112.100</address></Address>
</Node></Source><Target><Node>
<Address category="ipv4-addr"><address>202.77.162.213</address></Address>
</Node><Service><name>icmp-destination-unreachable</name></Service></Target>
</Alert></IDMEF-Message>
```

FIGURE 6.3 – exemple d'une alerte non pertinente

alertes générées. De ce cas, l'ensemble des faux négatifs ne peut pas être amélioré par le système proposé. Dans notre étude, le nombre de faux négatifs est le même que celui de Snort qui est évalué par 33814 [SFLZ14].

Afin de comparer notre approche à d'autres approches visant à réduire le nombre d'alertes non pertinentes et de faux positifs, nous utilisons l'approche d'Alireza et al [SFLZ14]. Les auteurs utilisent SWRL pour exprimer les règles d'inférence en prenant en considération le contexte des évènements survenus. Ils utilisent le premier scénario d'attaque LLDDOS 1.0 de l'ensemble de données DARPA 2000 pour l'évaluation.
Dans le tableau 6.8, le taux de faux positif est évalué par zéro % est justifié à cause que la liste des alertes déduites représente l'ensemble des hôtes qui satisfont les vulnérabilités exploitées. Parmis les alertes non sélectionnées, il y a un ensemble d'alertes qui correspondent à des actions qui préparent pour les actions détectées, ce qui signifier la nécessité d'une procédure qui permet de découvrir la relation entre les alertes sélectionnées et ceux qui n'ont pas. on peut observer que les résultats sont plus proches avec le même taux de faux positifs. Malgré cela, notre approche est plus dynamique vu sens que l'arrivage de nouvelles connaissances sur le contexte peut changer l'inférence immédiatement. Puisque cela revient à l'utilisation du raisonnement non monotone qui n'est pas fourni dans les systèmes bien connus comme les raisonneurs Racer, Fact ++ et Pellet.

Approche	**Faux positif**	**Faux négatif**	**Précision**	**Rappel**	**F-mesure**
Alireza et al	0%	33784	1	$3 * 10^{-3}$	$5 * 10^{-3}$
NOC-IDS	0%	33814	1	$2 * 10^{-3}$	$4 * 10^{-3}$

TABLE 6.8 – Comparaison des approches de réduction des faux positifs à l'aide du scénario d'attaque LLDDOS1.0

6.4 Conclusion

Dans ce chapitre, nous avons présenté les expérimentations de notre approche. L'approche est évaluée sur deux cas d'étude. Dans la première, nous avons utilisé une instance de l'attaque Island-hopping qui traite le changement de contexte avec une description de contexte sur les entités du réseau.

Ainsi, l'approche est évaluée sur le premier scénario LLDDOS1.0 de la base de données DARPA 2000.

Les résultats obtenus sont comparés avec d'autres approches dans le domaine de la détection d'intrusion qui traite le problème de faux positif en se basant sur les informations contextuelles.

A la fin, nous avons donné une discussion sur les résultats obtenus où notre modèle représente un outil efficace et complémentaire au système de détection d'intrusion avec la nécessité d'une procédure qui permet de découvrir la relation entre les alertes sélectionnées et celles que n'ont pas de sort que certaines alertes préparent pour les alertes déduites.

Conclusion Générale

Les systèmes de détection d'intrusions forment une ligne de défense indispensable pour chaque infrastructure. Son fonctionnement vise à analyser le trafic réseau et détecter les actions malicieuses. Il génère des messages qui contiennent des informations sur l'évènement détecté sachant l'adresse IP source, l'adresse IP destination, le temps de détection, la classification d'attaque, etc. Ces messages forment des alertes qui doivent être supervisées par un administrateur de sécurité.

Cependant, les systèmes de détection d'intrusion génèrent un nombre ingérable d'alertes qui inclut un taux de faux positifs très élevé. Ce fait oblige l'administrateur de sécurité à différencier entre les vraies attaques et celle de faux positifs.

Afin de remédier à ce problème, nous avons présenté dans cette thèse une étude sur une grande partie des travaux dont l'objectif est de réduire le nombre de faux positifs. Les travaux de la corrélation d'alerte peuvent être classés en quatre catégories : technique à base de similarité, technique à base de scénario d'attaque prédéfinies, technique à base de filtrage et des techniques de corrélation d'alerte multi-steps. On peut aussi ajouter une autre catégorie qui prend en considération des informations contextuelles pour interpréter les alertes générées.

La différence majeure entre ces travaux et l'approche proposée dans cette thèse est qu'elle traite le contexte et changement de contexte en profitant des avantages d'un raisonnement non monotone. Notre contribution peut être résumée comme suit :

-proposer une nouvelle représentation des informations contextuelles et répondre aux requêtes de pertinence.

-Identifier les alertes pertinentes et réduire le nombre des alertes non pertinentes.

-Générer une instance de l'attaque Island-hopping pour l'évaluation de cette approche. Cette évaluation permet d'évaluer la réduction des alertes faux positifs et d'autre part d'évaluer l'efficacité de traiter le changement de contexte.

L'approche est évaluée en utilisant une instance de l'attaque Island-hopping et le premier scénario LLDDOS1.0 de la base DARPA 2000. Les résultats obtenus prouvent que le modèle proposé dans cette thèse fonctionne comme un outil complémentaire au système de détection d'intrusion permettant de réduire le nombre d'alertes qui doivent être analysées par l'administrateur de sécurité, ainsi que de déduire les alertes pertinentes, avec la

nécessité d'une procédure qui permet de découvrir la relation entre les alertes sélectionnées et celles que n'ont pas de sort que certaines alertes préparent pour les alertes déduites.

Nous pouvons envisager différentes perspectives à ces travaux de recherche qui concerne notre approche. Bien que l'approche proposée dans cette thèse ne décrit pas touts les concepts pertinence de la détection d'intrusion et qui permettent un inférence plus fiable, nous visons à accomplir cette ontologie par d'autres concepts qui permettent de représenter les données de la détection d'intrusion et raisonner sur alertes pertinentes.

Dans le chapitre 6, nous avons proposé une règle d'inférence afin de détecter les stratégies des attaque à base des alertes pertinentes. Nous visons aussi à compléter la technique afin de reconstruire des scénarios d'attaques selon certaines relations entre les alertes.

Parmi les défauts de notre modèle est qu'il est incapable de détecter la relation entre les alertes afin de reconstruire les scénarios d'attaques.Ainsi l'approche ne traite pas le taux de faux négatifs. Bien que la pertinence des alertes est déterminée par le contexte des évènements survenus, pour éviter que certaines alertes non-sélectionnées peuvent être efficace à indiquer des intrusions et afin de réduire les faux négatifs, nous voyons d'associer une valeur de priorité aux alertes.

Une perspectif à long terme est de programmer des modules pour l'outil $TJClassic\delta\epsilon$, qui permettent de dessiner le graphe à partir des informations fournies par l'ontologie et faire la relation entre les concepts pour une meilleure représentation et une meilleure compréhension.

Bibliographie

[AAB12] Alsubhi, Khalid ; Aib, Issam ; Boutaba, Raouf : FuzMet : a fuzzy-logic based alert prioritization engine for intrusion detection systems. In : *Int. Journal of Network Management* 22 (2012), Nr. 4, 263–284. http://dx.doi.org/10.1002/nem.804. – DOI 10.1002/nem.804

[ADB+99] Abowd, Gregory D. ; Dey, Anind K. ; Brown, Peter J. ; Davies, Nigel ; Smith, Mark ; Steggles, Pete : Towards a better understanding of context and context-awareness. In : *Handheld and ubiquitous computing* Springer, 1999, S. 304–307

[ALJ08] Almgren, Magnus ; Lindqvist, Ulf ; Jonsson, Erland : A multi-sensor model to improve automated attack detection. In : *International Workshop on Recent Advances in Intrusion Detection* Springer, 2008, S. 291–310

[Als16] Alserhani, Faeiz M. : Alert correlation and aggregation techniques for reduction of security alerts and detection of multistage attack. In : *International Journal of Advanced Studies in Computers, Science and Engineering* 5 (2016), Nr. 2, S. 1

[BDP97] Benferhat, Salem ; Dubois, Didier ; Prade, Henri : Nonmonotonic reasoning, conditional objects and possibility theory. In : *Artificial Intelligence* 92 (1997), Nr. 1-2, S. 259–276

[BH95] Baader, Franz ; Hollunder, Bernhard : Embedding defaults into terminological knowledge representation formalisms Springer, 1995, S. 149–180

[BKM08] Benferhat, Salem ; Kenaza, Tayeb ; Mokhtari, Aicha : A naive bayes approach for detecting coordinated attacks. In : *Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International* IEEE, 2008, S. 704–709

[BM10] Boustia, Narhimene ; Mokhtari, Aicha : A contextual multilevel access control model with default and exception description logic. In : *Internet Technology and Secured Transactions (ICITST), 2010 International Conference for* IEEE, 2010, S. 1–6

[BM12] Boustia, Narhimene ; Mokhtari, Aicha : A dynamic access control model. In : *Applied Intelligence* 36 (2012), Nr. 1, S. 190–207

[BMPS+91] Brachman, Ronald J. ; McGuinness, Deborah L. ; Patel-Schneider, Peter F. ; Resnick, Lori A. ; Borgida, Alexander : Living with CLASSIC : When and how to use a KL-ONE-like language, 1991, S. 401–456

[Bou11] Boustia, Narhimene : *Un formalisme de sécurisation des bases de données multi-niveaux*, Université des Sciences et de la Technologie Houari Boumediene, PhD thesis, Diss., 2011

[CAI97] *CAIDA : Center for Applied Internet Data Analysis.* www.caida.org, 1997

[cap] *CAPEC : Common attack pattern enumeration and classification.* http ://capec.mitre.org/,

[CB16] Chergui, Nadjah ; Boustia, Narhimene : Using Vulnerability to Reduce False Positive Rate in Intrusion Detection Systems. In : *International Journal of Computer and Information Engineering* 10 (2016), Nr. 3, S. 486–491

[CB20] Chergui, Nadjah ; Boustia, Narhimene : Contextual-based approach to reduce false positives. In : *IET Information Security* 14 (2020), Nr. 1, S. 89–98

[CD00] Cordier, Marie-Odile ; Dousson, Christophe : Alarm driven monitoring based on chronicles. In : *IFAC Proceedings Volumes, Elsevier* 33 (2000), Nr. 11, S. 291–296

[CDER09] Coppolino, Luigi ; D'Antonio, Salvatore ; Elia, Ivano A. ; Romano, Luigi : From intrusion detection to intrusion detection and diagnosis : An ontology-based approach. In : *Software Technologies for Embedded and Ubiquitous Systems*. Springer, 2009, S. 192–202

[CF97] Coupey, Pascal ; Fouqueré, Christophe : Extending conceptual definitions with default knowledge. In : *Computational Intelligence, Wiley Online Library* 13 (1997), Nr. 2, S. 258–299

[CGHO16] Chen, Chia-Mei ; Guan, Dah-Jyh ; Huang, Yu-Zhi ; Ou, Ya-Hui : Anomaly network intrusion detection using hidden Markov model. In : *Int. J. Innov. Comput. Inform. Control* 12 (2016), S. 569–580

[CM02] Cuppens, Frédéric ; Miege, Alexandre : Alert correlation in a cooperative intrusion detection framework. In : *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on* IEEE, 2002, S. 202–215

[Cup01] Cuppens, Frédéric : Managing alerts in a multi-intrusion detection environment. In : *acsac* IEEE, 2001, S. 0022

[cve] *CVE : Common vulnerabilities exposures, the key to information sharing.* http ://cve.mitre.org/,

[Dan03] Dan, Gorton : Extending intrusion detection with alert correlation and intrusion tolerance. In : *Licentiate thesis,Chalmers University of Technology* (2003)

[DD08] Debarb, Benjamin Morina Ludovic Méa H. ; Ducassé, Mireille : M4D4 : a Logical Framework to Support Alert Correlation in Intrusion Detection. (2008)

[DDW99] Debar, Hervé ; Dacier, Marc ; Wespi, Andreas : Towards a taxonomy of intrusion-detection systems. In : *Computer networks, Elsevier* 31 (1999), Nr. 8, S. 805–822

[DDW00] Debar, Hervé ; Dacier, Marc ; Wespi, Andreas : A revised taxonomy for intrusion-detection systems. In : *Annales des télécommunications* Bd. 55 Springer, 2000, S. 361–378

[DEF] *DEF CON CTF Archive.* https ://www.defcon.org/html/links/dc-ctf-history.html,

[DNT04] Denker, Grit ; Nguyen, Son ; Ton, Andrew : Owl-s semantics of security web services : A case study. In : *European Semantic Web Symposium* Springer, 2004, S. 240–253

[DTEK09] Dao-Tran, Minh ; Eiter, Thomas ; Krennwallner, Thomas : Realizing default logic over description logic knowledge bases. In : *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty* Springer, 2009, S. 602–613

[ER08] El Rab, Mohammed G. : *Evaluation des systèmes de détection d'intrusion*, Université Paul Sabatier-Toulouse III, PhD thesis, Diss., 2008

[GME09] Gagnon, François ; Massicotte, Frédéric ; Esfandiari, Babak : Using contextual information for ids alarm classification. In : *Detection of Intrusions and Malware, and Vulnerability Assessment.* Springer, 2009, S. 147–156

[Goe16] Goeschel, Kathleen : Reducing false positives in intrusion detection systems using data-mining techniques utilizing support vector machines, decision trees, and naive Bayes for off-line analysis. In : *SoutheastCon 2016*, 2016, S. 1–6

[HHS+17] Haider, Waqas ; Hu, Jiankun ; Slay, Jill ; Turnbull, Benjamin P. ; Xie, Yi : Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling. In : *Journal of Network and Computer Applications, Elsevier* 87 (2017), S. 185–192

[HSH19] Hostiadi, Dandy P. ; Susila, Made D. ; Huizen, Roy R. : A new alert correlation model based on similarity approach. In : *2019 1st International Conference on Cybernetics and Intelligent System (ICORIS)* Bd. 1 IEEE, 2019, S. 133–137

[ICLC09] Isaza, Gustavo ; Castillo, Andrés ; López, Manuel ; Castillo, Luis : Towards ontology-based intelligent model for intrusion detection and prevention. In : *Computational Intelligence in Security for Information Systems*. Springer, 2009, S. 109–116

[KDD] *Knowledge discovery in databases DARPA archive. Task Description.* http ://www.kdd.ics.uci.edu/databases/kddcup99/task.html,

[Ken11] Kenaza, Tayeb : *Modèles graphiques probabilistes pour la corrélation d'alertes en détection d'intrusions*, Artois university et Université des Sciences et de la Technologie Houari Boumediene, PhD thesis, Diss., 2011

[KG18] Khalida Guesmia, Narhimene B. : *UN FORMALISME DE SÉCURITÉ DANS LE CLOUD COMPUTING*, Université Saad Dahlab Blida 1, Diss., 2018. – unpublished thesis

[KGWB15] Khan, Suleman ; Gani, Abdullah ; Wahab, Ainuddin Wahid A. ; Bagiwa, Mustapha A. : SIDNFF : Source identification network forensics framework for cloud computing. In : *Consumer Electronics-Taiwan (ICCE-TW), 2015 IEEE International Conference on* IEEE, 2015, S. 418–419

[KLK05] Kim, Anya ; Luo, Jim ; Kang, Myong : Security ontology for annotating resources. In : *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"* Springer, 2005, S. 1483–1499

[KS12] Klüft, Sebastian ; Staaf, Eva L. : Alarm management for intrusion detection systems-Prioritizing and presenting alarms from intrusion detection systems. In : *Computer Science Programme, University of Gothenburg, Sweden* (2012)

[LBN] *LBNL/ICSI Enterprise Tracing Project.* http ://www.icir.org,

[LHT18] Lanoe, David ; Hurfin, Michel ; Totel, Eric : A Scalable and Efficient Correlation Engine to Detect Multi-step Attacks in Distributed Systems. In :

37th IEEE International Symposium on Reliable Distributed Systems (SRDS 2018), 2018

[LQ05] Lee, Wenke ; Qin, Xinzhou : Statistical causality analysis of infosec alert data. In : *Managing Cyber Threats*. Springer, 2005, S. 101–127

[LT10] Li, Wan ; Tian, Shengfeng : An ontology-based intrusion alerts correlation system. In : *Expert Systems with Applications, Elsevier* 37 (2010), Nr. 10, S. 7138–7146

[MAJ13] Mirheidari, Seyed A. ; Arshad, Sajjad ; Jalili, Rasool : Alert correlation algorithms : A survey and taxonomy. In : *Cyberspace Safety and Security*. Springer, 2013, S. 183–197

[MBM15] Małowidzki, Marek ; Berezinski, P ; Mazur, Michał : Network intrusion detection : Half a kingdom for a good dataset. In : *Proceedings of NATO STO SAS-139 Workshop, Portugal*, 2015

[MCA00] McHugh, John ; Christie, Alan ; Allen, Julia : Defending yourself : The role of intrusion detection systems. In : *IEEE software* 17 (2000), Nr. 5, S. 42–51

[MD03] Morin, Benjamin ; Debar, Hervé : Correlation of intrusion symptoms : an application of chronicles. In : *International Workshop on Recent Advances in Intrusion Detection* Springer, 2003, S. 94–112

[MMDD09] Morin, Benjamin ; Mé, Ludovic ; Debar, Hervé ; Ducassé, Mireille : A logic-based model to support alert correlation in intrusion detection. In : *Information Fusion, Elsevier* 10 (2009), Nr. 4, S. 285–299

[MSM06] Martimiano, Luciana Andréia F. ; Santos Moreira, Edson dos : The Evaluation Process of a Computer Security Incident Ontology. In : *2nd Workshop on Ontologies and their Applications (WONTO)*, 2006

[NCR02] Ning, Peng ; Cui, Yun ; Reeves, Douglas S. : Constructing attack scenarios through correlation of intrusion alerts. In : *Proceedings of the 9th ACM conference on Computer and communications security* ACM, 2002, S. 245–254

[nvd] *NVD : Natinal vulnerability database (NVD), automatin vulnerability management, security measurement, and compliance checking*. http ://nvd.nist.gov/,

[PFV02] Porras, Phillip A. ; Fong, Martin W. ; Valdes, Alfonso : A mission-impact-based approach to INFOSEC alarm correlation. In : *International Workshop on Recent Advances in Intrusion Detection* Springer, 2002, S. 95–114

[PN93] Padgham, Lin ; Nebel, Bernhard : Combining classification and nonmonotonic inheritance reasoning : A first step. In : Komorowski, Jan (Hrsg.) ; Raś, Zbigniew W. (Hrsg.) : *Methodologies for Intelligent Systems*. Berlin, Heidelberg : Springer Berlin Heidelberg, 1993, S. 132–141

[PS03] Powell, David ; Stroud, Robert : Conceptual Model and Architecture for MAFTIA. In : *School of Computing Science Technical Report Series* (2003)

[PZ93] Padgham, Lin ; Zhang, Tingting : A terminological logic with defaults : A definition and an application. In : *IJCAI* Bd. 93, 1993, S. 662–668

[QR92] Quantz, Joachim ; Royer, Veronique : A Preference Semantics for Defaults in Terminological Logics. In : *KR* 92 (1992), S. 294–305

[R93] R, Gruber T. : A translation approach to portable ontology specifications. In : *Knowledge acquisition, Elsevier* 5 (1993), Nr. 2, S. 199–220

[RAA19] Riyad, Ali M. ; Ahmed, Mohammed Saleem I. ; Almistarihi, Husni H. : A Quality Framework to Improve IDS Performance Through Alert Post-Processing. In : *International journal of Intelligent Engineering Systems* (2019)

[Rei80] Reiter, Raymond : A logic for default reasoning. In : *Artificial intelligence, Elsevier* 13 (1980), Nr. 1-2, S. 81–132

[RHTN01] Raskin, Victor ; Hempelmann, Christian F. ; Triezenberg, Katrina E. ; Nirenburg, Sergei : Ontology in information security : a useful theoretical foundation and methodological tool. In : *Proceedings of the 2001 workshop on New security paradigms* ACM, 2001, S. 53–59

[SFLZ14] Sadighian, Alireza ; Fernandez, José M ; Lemay, Antoine ; Zargar, Saman T. : ONTIDS : A Highly Flexible Context-Aware and Ontology-Based Alert Correlation Framework. In : *Foundations and Practice of Security*. Springer, 2014, S. 161–177

[SH05] Seacord, Robert C. ; Householder, Allen D. : A structured approach to classifying security vulnerabilities / carnegie-mellon univ pittsburgh pa software engineering inst. 2005. – Forschungsbericht

[SM12] Scarfone, Karen ; Mell, Peter : Guide to intrusion detection and prevention systems (idps) / National Institute of Standards and Technology. 2012. – Forschungsbericht

[SP10] Sommer, Robin ; Paxson, Vern : Outside the closed world : On using machine learning for network intrusion detection. In : *Security and Privacy (SP), 2010 IEEE Symposium on* IEEE, 2010, S. 305–316

[SS14] Sherif Saad, Marcelo Luiz B. Issa Traore T. Issa Traore : Context-aware intrusion alerts verification approach. In : *Information Assurance and Security (IAS), 2014 10th International Conference on. Florida : IEEE* (2014)

[SSTG12] Shiravi, Ali ; Shiravi, Hadi ; Tavallaee, Mahbod ; Ghorbani, Ali A. : Toward developing a systematic approach to generate benchmark datasets for intrusion detection. In : *computers & security, Elsevier* 31 (2012), Nr. 3, S. 357–374

[UJP03] Undercoffer, Jeffrey ; Joshi, Anupam ; Pinkston, John : Modeling computer attacks : An ontology for intrusion detection. In : *Recent Advances in Intrusion Detection* Springer, 2003, S. 113–135

[VB+10] Vorobiev, Artem ; Bekmamedova, Nargiza u. a. : An ontology-driven approach applied to information security. In : *Journal of Research and Practice in Information Technology* 42 (2010), Nr. 1, S. 61

[VS01] Valdes, Alfonso ; Skinner, Keith : Probabilistic alert correlation. In : *Recent advances in intrusion detection* Springer, 2001, S. 54–68

[WG09] Wang, Ju A. ; Guo, Minzhe : OVM : an ontology for vulnerability management. In : *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research : Cyber Security and Information Intelligence Challenges and Strategies* ACM, 2009, S. 34

[WSP08] Wu, Handong ; Schwab, Stephen ; Peckham, Robert L. : *Signature based network intrusion detection system and method.* September 9 2008. – US Patent 7,424,744

[XN05] Xu, Dingbang ; Ning, Peng : Privacy-preserving alert correlation : a concept hierarchy based approach. In : *Computer Security Applications Conference, 21st Annual* IEEE, 2005, S. 10–pp

[XTT05] Xu, Heng ; Teo, Hock-Hai ; Tan, Bernard : Predicting the adoption of location-based services : The role of trust and perceived privacy risk. In : *ICIS 2005 proceedings* (2005), S. 71

[YÇ10] Yue, Wei T. ; Çakanyildirim, Metin : A cost-based analysis of intrusion detection system configuration under active or passive response. In : *Decision Support System, Elsevier* 50 (2010), Nr. 1, S. 21–31

[YSS08] Yusof, Robiah ; Selamat, Siti R. ; Sahib, Shahrin : Intrusion alert correlation technique analysis for heterogeneous log. In : *International Journal of Computer Science and Network Security* 8 (2008), Nr. 9, S. 132–138

[ZLK10] Zhou, Chenfeng V. ; Leckie, Christopher ; Karunasekera, Shanika : A survey of coordinated attacks and collaborative intrusion detection. In : *Computers & Security, Elsevier* 29 (2010), Nr. 1, S. 124–140